

AD-A200 406

A TRIDENT SCHOLAR PROJECT REPORT

NO. 149

DTIC FILE COPY

Multiple Fault Diagnosis System



UNITED STATES NAVAL ACADEMY
ANNAPOLIS, MARYLAND

This document has been approved for public
release and sale; its distribution is unlimited.

DTIC
ELECTE
NOV 03 1988
S H D

88 11 3 053

UNCLASSIFIED

SECURITY CLASSIFICATION OF THIS PAGE (When Data Entered)

REPORT DOCUMENTATION PAGE		READ INSTRUCTIONS BEFORE COMPLETING FORM
1. REPORT NUMBER U.S.N.A. - TSPR; no. 149 (1988)	2. GOVT ACCESSION NO.	3. RECIPIENT'S CATALOG NUMBER
4. TITLE (and Subtitle) MULTIPLE FAULT DIAGNOSIS SYSTEM.		5. TYPE OF REPORT & PERIOD COVERED Final 1987/88
		6. PERFORMING ORG. REPORT NUMBER
7. AUTHOR(s) Bryan Paul Graham		8. CONTRACT OR GRANT NUMBER(s)
9. PERFORMING ORGANIZATION NAME AND ADDRESS United States Naval Academy, Annapolis.		10. PROGRAM ELEMENT, PROJECT, TASK AREA & WORK UNIT NUMBERS
11. CONTROLLING OFFICE NAME AND ADDRESS United States Naval Academy, Annapolis.		12. REPORT DATE 10 June 1988
		13. NUMBER OF PAGES 81
14. MONITORING AGENCY NAME & ADDRESS (if different from Controlling Office)		15. SECURITY CLASS. (of this report)
		15a. DECLASSIFICATION/DOWNGRADING SCHEDULE
16. DISTRIBUTION STATEMENT (of this Report) This document has been approved for public release; its distribution is UNLIMITED		
17. DISTRIBUTION STATEMENT (of the abstract entered in Block 20, if different from Report)		
18. SUPPLEMENTARY NOTES Accepted by the U.S. Trident Scholar Committee.		
19. KEY WORDS (Continue on reverse side if necessary and identify by block number) Expert systems (Computer science) Systems design		
20. ABSTRACT (Continue on reverse side if necessary and identify by block number) The Navy currently uses Automated Test Systems (ATEs) to diagnose faults in most operational hardware units. There are numerous problems associated with the use of ATEs. The problems that can be solved through the application of improved technology involve high removal rate of good components, excessive levels of fault ambiguity, and lack of a successful diagnosis. Inherent in the use of ATEs are the two problems which render ATE use inefficient. (OVER)		

DD FORM 1473
1 JAN 73EDITION OF 1 NOV 65 IS OBSOLETE
S N 0102-LF-014-6601

UNCLASSIFIED

SECURITY CLASSIFICATION OF THIS PAGE (When Data Entered)

UNCLASSIFIED

SECURITY CLASSIFICATION OF THIS PAGE (When Data Entered)

They are expensive and highly specialized. A single multi-million dollar test unit may only test a single hardware unit. One way to solve this problem is to develop a single expert system which is knowledgeable about circuit operation in general and is thus able to diagnose faults in many different hardware units. It will be able to make suggestions, in an interactive manner with the technician, concerning the location of the next best test utilizing information from an a priori failure rate database. Through a series of tests on the unit under test (UUT), the expert system will correctly isolate the fault.

The objective of this project was to develop a versatile, interactive prototype of a fault diagnosis expert system capable of diagnosing multiple faults in generic electronic component systems and to test this system over several actual hardware setups capable of being faulted and subsequently tested.

The prototype system was developed using GOLDWORKS, an AI development shell from Gold Hill Computers. This shell provided a frame-based development. The working system was tested on several hardware setups constructed on Hewlett-Packard 5035T Logic Lab breadboards. The resulting multiple fault diagnosis system is a powerful tool to aid an electronics technician in diagnosing faults in a digital circuit. It allows the technician not only to interactively observe the model and test the faulty unit, but also allows the system to make suggestions for the location of the next best test. The power of this system lies in the fact that it utilizes knowledge about how circuits operate and applies it to the diagnosis of many different types of circuits.



Accession For	
NTIS GRA&I	<input checked="" type="checkbox"/>
DTIC TAB	<input type="checkbox"/>
Unannounced	<input type="checkbox"/>
Justification	
By	
Distribution/	
Availability Codes	
Avail and/or	
Dist	Special
A-1	

S N 0102-LF-014-6601

UNCLASSIFIED

SECURITY CLASSIFICATION OF THIS PAGE (When Data Entered)

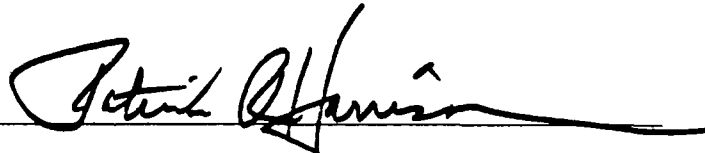
U.S.N.A. - Trident Scholar project report; no. 149 (1988)

Multiple Fault Diagnosis System
A Trident Scholar Project Report

by

Bryan Paul Graham
Midshipman First Class, 1988

United States Naval Academy
Annapolis, Maryland



Professor Harrison

Accepted for Trident Scholar Committee



Chairperson

10 June 1988

Date

USNA-1531-2

Multiple Fault Diagnosis System
Bryan P. Graham
United States Naval Academy

1. Background: The Navy currently uses Automated Test Equipment (ATEs) to diagnose faults in most operational hardware units. There are numerous problems associated with the use of ATEs. The problems that can be solved through the application of improved technology involve high removal rate of good components, excessive levels of fault ambiguity, and lack of a successful diagnosis. Inherent in the use of ATEs are two problems which render ATE use inefficient. They are expensive and highly specialized. A single multi-million dollar test unit may only test a single hardware unit. One way to solve this problem is to develop a single expert system which is knowledgeable about circuit operation in general and is thus able to diagnose faults in many different hardware units. It will be able to make suggestions, in an interactive manner with the technician, concerning the location of the next best test utilizing information from an a priori failure rate database. Through a series of tests on the unit under test (UUT), the expert system will correctly isolate the fault.

2. Objectives: The objective of this project was to develop a versatile, interactive prototype of a fault diagnosis expert system capable of diagnosing multiple faults in generic electronic component systems and to test this system over several actual hardware setups capable of being faulted and subsequently tested.

3. Methodology and Results: The prototype system was developed using GOLDWORKS, an AI development shell from Gold Hill Computers. This shell provided a frame-based development environment. The working system was tested on several hardware setups constructed on Hewlett-Packard 5035T Logic Lab breadboards. The resulting multiple fault diagnosis system is a powerful tool to aid an electronics technician in diagnosing faults in a digital circuit. It allows the technician not only to interactively observe the model and test the faulty unit, but also allows the system to make suggestions for the location of the next best test. The power of this system lies in the fact that it utilizes knowledge about how circuits operate and applies it to the diagnosis of many different types of circuits.

Table of Contents

1. INTRODUCTION
 - 1.1. Current use of ATE's to diagnose hardware problems
 - 1.2. Expert Systems as a diagnosis tool
 - 1.3. Objectives
2. DISCUSSION
 - 2.1. Principle of operation
 - 2.2. Constructing the model
 - 2.3. Importance of choice of level of abstraction as it affects locality and causal relationships
 - 2.4. Testing the outputs and creation of the suspect set
 - 2.5. Testing to reduce the suspect set in the single-fault case
 - 2.6. Complications in the multiple-fault case
 - 2.7. Isolating faults when the single fault constraint is relaxed
 - 2.8. Finding the best test through the computation of gain
3. SAMPLE DIAGNOSIS SESSIONS
 - 3.1. Single-fault case: 3-bit adder circuit
 - 3.2. Multiple-fault case: 5-bit comparator
4. CONCLUSION
5. ACKNOWLEDGMENTS
6. BIBLIOGRAPHY
7. FIGURES
8. APPENDIX A

List of Figures

1. Simple Adder
2. Cascaded Low-Pass Filter
3. "Black-Box" Filter
4. 3-Bit Adder
5. 5-Bit Comparator

1. INTRODUCTION

1.1. *Current use of ATE's to diagnose hardware problems*

Every time the Navy introduces a new piece of operational hardware into the Fleet, it must also provide millions of dollars worth of test equipment to support its use. The Navy currently uses Automated Test Equipment (ATE) to diagnose faults in most operational hardware units. There are numerous documented problems associated with the use of ATEs.

- o Not always successful
- o High removal rate of good components
- o Excessive levels of fault ambiguity
- o Prohibitive cost
- o Highly specialized to test only one unit

The first three of these problems can be solved through the application of improved technology, but the latter two problems are inherent in the use of ATEs. One way to solve these problems is to develop a single expert system which is able to diagnose faults in many different hardware units.

1.2. Expert systems as a diagnosis tool

One such expert system might be developed in an artificial intelligence environment using the LISP programming language. Ideally, it would diagnose faults in many different hardware units and would operate interactively with a technician. It would be able to make suggestions concerning the location of the next best test utilizing information from an a priori failure rate database. Through a series of tests on the unit under test (UUT), the expert system will correctly isolate the fault.

1.3. Objectives

The objectives of this project were to (1) develop a versatile, interactive prototype of a fault diagnosis expert system capable of diagnosing multiple faults in generic electronic component systems and (2) to test this system over several actual hardware setups capable of being faulted and subsequently tested. The system was to be developed in a LISP environment and be capable of implementation on a personal computer.

2. DISCUSSION

2.1. *Principle of operation*

Using model-based reasoning, two items of knowledge are necessary to correctly diagnose faults in electronic component systems; how a unit is supposed to operate and how that unit is actually operating. A model or simulation of the unit is built and the operation of that model is compared to the operation of the actual unit. From the statistical difference in the values of their operation, inferences can be made as to the location of the fault in the unit. This type of inferencing is termed *model-based reasoning*.

2.2. *Constructing the model*

Construction of a model involves knowledge concerning the operation and the connectivity of all components within the system. Describe each component as a transfer function which maps inputs to outputs. These outputs are connected to other components and values are propagated through the unit. An example to illustrate the construction of a model is the simple digital adder circuit shown in Figure 1.

Referring to the figure, A1 is an AND gate with inputs connected to C1 and C2 and output connected to C5. Assume the input values are 1 and 0 (ON and OFF in the digital sense.) The output value is then 0 arrived at from the function of an AND gate. We can describe its presence in the circuit, its connectivity and its state in the following manner:

```
(A1  (ANDGATE)
      (CON (C1 C2)
           (C5))
      (VAL (1 0))
      (OUT (0))
```

The transfer function (or the mapping of inputs to outputs) of a component can be similarly represented. In a simplified form the function of an AND gate can be described as:

```
(if A1 is ANDGATE with
    (VAL (0 0) or (1 0) or (0 1))
then (OUT (0)))
```

```
(if A1 is ANDGATE with
    (VAL (1 1))
then (OUT (1)))
```

In English this reads, if A1 is an ANDGATE and has the VALues (0 0) or (1 0) or (0 1), then its OUTput will be 0, OR if A1 is an ANDGATE with VALues (1 1), then its OUTput will be 1. What has been supplied to the model is a functional description of an AND gate which maps all possible inputs to outputs.

Similar functional and connective descriptions are supplied for each component in the circuit. If values are placed in the circuit board's input ports (I1 I2 I3), then those values can be propagated throughout the circuit using the connectivity and function descriptions of the components. All of this takes place within the model constructed in computer code.

The model provides a standard by which to compare and test the actual circuit board. Without a model, it would be nearly impossible to judge faulty circuit behavior properly. This is the basic precept of model-based reasoning.

2.3. Importance of choice of level of abstraction as it affects locality and causal relationships

The concepts of locality and causal reference are directly dependent upon the level of abstraction which is chosen to model electronic systems and strongly affects the

ability to model and diagnose faults in a system. If a component supports locality, then the causal relationships that stem from its operation affect only those components and values that are direct neighbors in the component system. For locality to occur, there must be a direct causal path that extends only to each component's immediate neighbor. This notion of locality is directly dependent upon our choice of level of abstraction of the system.

Due to the level of abstraction chosen to depict the cascaded low-pass filter in Figure 2, the concept of locality does not apply. The input to that system is V_s and its output is V_o taken across the capacitor C2. Locality does not apply because each component in the system not only affects the voltages and currents to its neighbors, but directly affects the values found throughout the circuit. There is no direct causal path from one component to another because they are all interdependent. For example, assume the resistor R1 shorts out; this short circuit raises the current through C2 and increases the output voltage V_o . Due to the interdependence of all components, the causal relationships involved in the operation of R1 extend past the direct neighbors, thus locality is not present.

The main problem associated with lack of locality is the difficulty in modeling such systems. The computational expense of applying Kirchoff's voltage and current laws, along with Ohm's law (for a computer), to the cascaded filter would be fairly large, again due to the interdependencies of all components involved. To solve this problem, we can choose a different level of abstraction with which to view the problem. In this example, transform the four components of the cascaded filter into a single "black box" as shown in Figure 3 with the same input and output. A single transfer function $G(V_S)$ takes the place of the applied rules that had previously modeled the circuit at a lower level of abstraction. Locality now applies to the system because the cascaded filter is only directly affected by its neighbor V_S and it only directly affects its neighbor V_O . This abstracted component may now be placed in a larger system while preserving locality of reference.

If a component's causal relations only affect the system through the influence of its neighbors, and locality applies, then diagnosis becomes a much easier task. If we try to diagnose the short-circuited capacitor, C_1 in Figure 2., we would eventually take measurements at each node in the circuit. This would not be a problem for this circuit because there are only four nodes which can be tested. As systems increase in size this approach would become unfeasible.

So, we would raise the level of abstraction to introduce locality to the problem and then begin diagnosis. The input voltage applied at V_s is correct, and the output voltage V_o is incorrect, thus the fault must lie within the cascaded filter. Obviously, we have lost something in the translation to a higher level of abstraction; we have lost the ability to diagnose faults of a finer granularity. But when placed in circuits of greater complexity and larger numbers of components, what is gained through locality is tremendous.

2.4. *Testing the outputs and creation of the suspect set*

Once the model has been either created or loaded from a central database, diagnosis may begin. The first step is to test outputs. This provides a direction for diagnosing the faulty component because the UUT itself can be thought of as a *black box*. Its inputs are mapped through the structure and function of each of its components to the unit's outputs. Thus, any fault occurring in an individual component will cause an incorrect output to appear in the UUT. The diagnosis system recognizes this fault when comparing the model's expected outputs to the UUT's observed output.

To test the outputs of the UUT, inputs must be supplied to both the model and the UUT. To test the unit fully, we must apply the inputs in a systematic manner. A digital circuit has a finite number of input sets and by testing the circuit over all of these inputs we can be certain that all faults will be isolated. Only through the application of all input sets, may fault coverage be achieved in a digital circuit.

Once the input set has been applied to both the model and the UUT circuit inputs, we must test each output of the actual circuit for correctness. UUT output values are compared with the model output values. From the knowledge of the outcome of the test (OK/BAD), a hypothesis is formulated concerning the possible cause of the fault.

A fault hypothesis takes the form of a suspect set. A suspect set is the set of components that could possibly be responsible for a given fault in the circuit's output. Structurally, it is the set of components that uniquely determine a value at the faulty output (the components that were responsible for propagating a value to the particular output or its source list).

Again consider the adder example in Figure 1. If the output at S returned a faulty value then the suspect set created would be (R1 R2) because these two gates alone generate the output at S. If C were to be faulty, then the suspect set would appear as (R1 A1 A2 O1). Logically, if both outputs of the UUT generated faulty values, then all components in the circuit would be suspect.

A component's absence from the suspect set does not, however, assure its correct operation, only its innocence in explaining a particular fault of the outputs. For example, in the adder circuit suppose that we found the output at O-1 to be faulty and the suspect set generated was (X1 X2). The AND gate A2 is not present in the suspect set, but assume it is malfunctioning. Its faulty output might be masked from the output O-2 by a value of 1 at C5 going into the OR gate O1. The output at O-2 is still correct in spite of the fact that A2 is misbehaving.

This presents a difficult problem inasmuch as a component might be faulty and not be present in the suspect set. To correct this special case, apply all possible inputs to the circuit and assume that one particular input set will provide the values necessary to place the faulty component in the suspect set. Thus, fault coverage has been

achieved by testing the circuit over all possible inputs rather than trying to do so by expanding the suspect set to cover faults not directly causing a faulty output.

It is important to remember that the suspect set is formed from the set of components that could possibly explain a given faulty output. Components that might be faulty, but do not cause an incorrect output, are included. It is therefore possible to have a unique suspect set for each set of inputs.

2.5. Testing to reduce the suspect set in the single-fault case

After creation of the suspect set, the next step is to reduce this suspect set to a single component to fully isolate the fault. To accomplish this end, tests must be made and knowledge collected on the actual operation of the UUT. Test results from the UUT are compared to the values derived from the model. Discrepancy or agreement between these two values yields knowledge concerning UUT component operation. Thus, through a series of tests, individual component correctness or faultiness can be validated and therefore removed from or asserted to the suspect set. To this end, the knowledge obtained from UUT tests is ultimately aimed at

reducing the suspect set to a single component. That component is the sole source of the faulty values observed at the board's output ports.

Consider a series of tests on the 5-Bit Comparator in Figure 5. We have previously tested the outputs, L2, L3, and L4 and the results of those tests were (L2 OK), (L3 BAD), (L4 BAD). Our suspect set now contains all components in the circuit. We first choose to perform a test at node C16. (This may or may not be a good choice for a test, as will be discussed later.)

Assume the test performed at C16 returns an incorrect value. We now know two things, (1) that there is only one fault, (we made that constraint to begin with), and (2) the faulty component propagates its improper value through C16 on its way on the board output. Therefore, the fault must lie somewhere in the components responsible for propagating values to that node. In other words, the source-list for C16. This source-list is: (N5 V6 A1 N4 V3 N2 V2 N1 V1). All other components can then be removed from the suspect set. Those components which have been removed then constitute the reduction set resulting from the bad test, or RS_{bad} .

The occurrence of a good test is somewhat more complicated. If the test at C16 returns a correct value then the same reasoning doesn't follow that all modules in that node's source-list are assumed to be operating correctly. Only the component that is directly responsible for the value at the test node may be immediately removed from the suspect set. This is because it alone is not creating a fault which could be propagated through the test node to the outputs. To illustrate the reasoning behind this more clearly, assume the AND gate A1 is malfunctioning in our 5-Bit Comparator. This malfunction causes a faulty value at C11 and through C20 it is propagated to the outputs, but because of the truth-table associated with the three-input NAND gate N5, the fault is "masked" from appearing at C16.

Superficially, this poses a particularly difficult problem with regard to reduction of the suspect set: a good test will only remove one component from the suspect set, because a good test does not necessarily validate the correct operation of that node's source-list. Upon further investigation, however, we see that this is not truly the case. Again, back to our example of a good first test at C16. We can immediately remove N5 from the suspect set. Now consider other components such as the inverter V6. If V6 were to fault, and propagate that fault to the outputs, the fault must be propagated through the node C16. Therefore,

if a good test were conducted at C16, then the set of components removed from the suspect set, resulting from a good test or RS_{ok} are (N5 V6).

But why then can't we remove A1 as well? What is the difference between A1 and V6? They are both members of C16's source-list. The difference lies in the fact that a fault at V6 must propagate through C16 to the board outputs, i.e. we can say that it is a positively unmasked parent of C16. When we examine A1, we see that if it is faulty, its faults may be passed to both N5 and A4 and may be masked by those gates operation at either C16 or C20. Therefore, A1 is a possibly masked parent of C16 and C20. Only through testing and finding correct values at both C16 and C20, may we remove A1 from the suspect set.

It would now seem that a good test is relatively weak compared to a bad test in terms of reducing the suspect set. This however, is not truly the case. While it is true that a randomly chosen test will generally have only one or two members in its RS_{ok} , two or more good tests performed in conjunction can be fairly powerful in reducing the suspect set. Again assume that all components in the comparator are suspect and our first chosen test is C25. The node C25 has an RS_{ok} of (A6 N8 V7) and it returns a good value. Those components are removed from the suspect set and the

next test chosen is C26. Taken alone, the node C25 has an RS_{OK} of (A5 N7 V8), but since a test at C25 has already returned a value of OK, and all faults generated on the left side of C25 and C26 must pass through those nodes to reach the outputs, a good test at C26 will remove all components to the left of C25 and C26.

Thus, certain tests when chosen correctly singly, or in combination with other tests, can work to reduce the suspect set significantly when either a correct or incorrect value is observed.

2.6. Complications in the multiple fault case

Testing to reduce the suspect set in the single fault case is a relatively simple and straightforward process. When multiple faults are introduced, the process becomes much more complicated. The main difficulty in diagnosing a multiple fault case lies in determining what test results actually reveal concerning the UUT's true operation.

As in a single fault case, an OK test does not guarantee the correct operation of all components in that test's source list, but as the single-fault constraint is relaxed, a BAD test does not rule out the chance that a fault might lie outside of the test's source list. In other

words, when a BAD test occurs the components that are not in the test's source list cannot be removed from the suspect set because they might also be faulty.

There are several approaches to diagnosis in the multiple fault case. Two of note are Belief Revision in Bayesian Networks using Message Passing (Geffner and Pearl 1987), and Minimal Entropy Modeling. These two were found to be extremely complicated and computationally expensive. The approach used in this project was to isolate faults singly, correct the faults and then proceed to diagnose the other faults present in the system. Part of the information from each diagnosis acted to constrain the search for subsequent faults.

2.7. Isolating faults when the single fault constraint is relaxed

The single fault constraint we placed on our diagnosis earlier was in itself, quite a powerful tool for reducing the suspect set. When that constraint is no longer available, the diagnosis becomes more difficult by at least an order of magnitude. The approach taken in this project is to use the power of the single fault constraint to diagnose a fault, correct that fault, and then in essence begin again diagnosing the circuit.

Returning to our 5-bit comparator, assume that both V8 and V11 are faulty. Proceeding with our usual diagnosis, we make tests at ((C23 BAD) (C11 OK) (C12 OK) (C17 OK) (C21 BAD)). At this point we have correctly diagnosed the fault located at V8, but we have not addressed the fault located at V11. We then correct the fault at V8, retest the outputs and begin again. It would seem a waste to completely throw out all knowledge we already have gained from the circuit; perhaps some of it still holds true. The tests made at C11, C12 and C17 are good and lie upstream from the fault diagnosed at V8, thus the components that were removed from the suspect set as a result of those tests cannot possibly be responsible for any other faults with respect to this input set.

Assume that while trying to diagnose V8 we also took a test at C30 and it returned OK, can we immediately validate the correct operation of N10 and V9 for the subsequent diagnosis? The answer is no. If a fault were to lie at V9 (remember, we do not know how many faults there are in the system,) then the OK value found at C30 might be changed to an incorrect value when the fault at V8 is corrected. This is because C30 lies downstream from V8. Thus, any test that previously returned an OK value and lies outside the values generated by the recently corrected faulty component may be

used to immediately reduce the suspect set. The diagnosis proceeds accordingly until a correct output is observed for the entire system.

2.8. *Finding the best test through the computation of gain*

An important feature of our expert system should be the ability to suggest the next best test for the isolation of the fault. One way to locate the best location for the next test is through the computation of an expected gain associated with each possible test in the system. This gain is based upon how a test might reduce the suspect set and is weighted according to the failure rates of their respective reduction sets.

It would seem that the greater a test's reduction set, then the faster a suspect set could be reduced. However, this is not the case if the test returns a correct value. Thus, the best test is one which would reduce the suspect set by exactly one-half. The major drawback here is that we haven't taken into account any type of past failure rate knowledge.

To correct this shortcoming, the size of the reduction set of a test can be weighted according to the a priori failure rates of the components within that set. This is

termed the gain of a set. Mathematically, a gain is the sum of failure rates of a given set of components. To speak of the gain of a test, then, means to take the gain of that test's reduction set. The optimal gain for a test is one half of the sum of the failure rates of the suspect set (or gain of the suspect set). Thus, the result of a test will optimally reduce the suspect set by one half and weighed toward one half or another according to the component failure rates associated with each test's reduction set.

Again consider the adder circuit in Figure 1. We have already tested the outputs of the circuit and have found the output at O-1 to be correct, and the output at O-2 to be faulty. We have then validated the operation of A1 by a correct test at C5. Thus our suspect set is (X1 A2 O1). The set of possible tests that are relevant to the suspect set is (C1 C2 C3 C4 C6 C8). Of these tests only C4, C6, and C8 reduce the suspect set (i.e. they have reduction sets). Their reduction sets are:

C4: (X1)

C6: (X1 A2)

C8: (X1 A2 O1)

Each of these tests will perform useful work on the suspect set. We must choose the test which will reduce the suspect set regardless of the result of that test (OK/BAD) and also take into account the past failure rates of each component in the suspect set. To do so, we look into our a priori failure rate database and find the failure rates of these components to be:

XOR Gate: .025

AND Gate: .010

OR Gate: .015

These failure rates represent the actual field-tested a priori failure rate of each type of component. To find the optimal gain for the next test, we sum the failure rates of all components in the suspect set and divide by 2.

$$\text{Optimal Gain} = (.025 + .010 + .015) / 2 = .025$$

Now we wish to choose the test whose gain is closest to this optimal value. Taking the gains of each test:

$$C4 = .025$$

$$C6 = .025 + .010 = .035$$

$$C8 = .025 + .010 + .015 = .050$$

A test at C4 is closest to optimal. Logically this follows because the a priori failure rate for XOR gates is greater than the sum of the failure rates of the other two. If the test at C4 is found to be bad (statistically, this will happen more often than not) we have found our fault in component X1. Otherwise, we narrow the suspect set to two components, A2 and O1.

What is accomplished by choosing the next test according to computed gain is two-fold: it chooses the test that will reduce the suspect set most efficiently and also isolates the fault that occurs more frequently.

3. SAMPLE DIAGNOSIS SESSIONS ON TWO DIGITAL CIRCUITS

3.1 Single-Fault Case: 3-Bit adder circuit in Figure 4.

FAULT: R3

INPUT SET: ((X3 1)(X2 0)(X1 1)(Y3 0)(Y2 0)(Y1 1))

Test Outputs: ((C BAD)(S3 BAD)(S2 BAD)(S1 OK)

Suspect Set: (R1 A1 A2 O1 R3 R4 A3 A4 O2 R5 R6 A5 A6

O3)

Best Test: C13

Make Test #1: (C13 BAD) Remove (R4 A3 O2 R5 R6 A5 A6)

Suspect Set: (R1 A1 A2 O1 R3 A4)

Best Test: C6

Make Test #2: (C6 OK) Remove (R1 A2)

Suspect Set: (A1 O1 R3 A4)

Best Test: C10

Make Test #3: (C10 OK) Remove (A1 O1)

Suspect Set: (R3 A4)

Best Test: C9

Make Test #4: (C9 BAD) Remove (A4)

Suspect Set: (R3) -- FAULT DIAGNOSED!

3.2 Multiple-Fault Case: 5-Bit comparator in Figure 5.

FAULT: (V8 V11)

Test Outputs: ((L2 BAD) (L3 OK) (L4 BAD))

Suspect Set: (All components except R1)

Best Test: C23

Make Test #1: (C23 BAD)

Remove (V10 V7 V6 N5 A3 N8 A6 N9 A5 A8 V12 V9 N10 A7

V11)

Suspect Set: (V2 V3 N2 N4 V1 A1 V4 N3 A2 V5 N6 A4 V8

N7)

Best Test: C11

Make Test #2: (C11 OK) Remove (A1 N4 V2 V3)

Suspect Set: (V1 N1 V4 N3 A2 V5 N6 A4 V8 N7)

Best Test: C12

Make Test #3: (C12 OK) Remove (A2 N3 V4 N1 V1)

Suspect Set: (V5 N6 A4 V8 N7)

Best Test: C17

Make Test #4: (C17 OK) Remove (V5 N6)

Suspect Set: (A4 V8 N7)

Best Test: C21

Make Test #5: (C21 BAD) Remove (A4 N7)

Suspect Set: (V8) -- FIRST FAULT DIAGNOSED!

Correct Fault at V8.

Test Outputs: ((L2 OK) (L3 BAD) (L4 OK))

Tests: ((C17 OK) (C12 OK) (C11 OK)) Still hold true.

Suspect Set: (R1 V11 A7 N10 V9 V12 A8 A5 N7 V8 N9 A6
A4 N8 A3 V10 V11)

Best Test: C29

Make Test #6: (C29 OK) Remove (V10 N9)

Suspect Set: (R1 V11 A7 N10 V9 V12 A8 A5 N7 V8 A6 A4
N8 A3 V11)

Best Test: C25

Make Test #8: (C25 OK) Remove (V7 N8 A6)

Suspect Set: (R1 V11 A7 V12 A8 A5 N7 V8 V6 A3 N5 A4)

Best Test: C26

Make Test #9: (C26 OK) Remove (V6 N5 A3 A4 V8 N7 A5)

Suspect Set: (R1 V11 A7 V12 A8)

Best Test: C34

Make Test #10: (C34 OK) Remove (A8 V12)

Suspect Set: (R1 V11 A7)

Best Test: C33

Make Test #11: (C33 BAD) Remove (R1)

Suspect Set: (V11 A7)

Best Test: C31

Make Test #12: (C31 OK) Remove (A7)

Suspect Set: (V11) -- SECOND FAULT DIAGNOSED!

4. CONCLUSION

Toward the completion of the stated objectives of this Trident Scholar research project, the prototype system was developed using GOLDWORKS, an AI development shell from Gold Hill Computers, which provided a frame-based development environment. The working system was then tested on several hardware setups constructed on Hewlett-Packard 5035T Logic Lab breadboards. The code which constructed the framework of the system can be seen in Appendix A.

The multiple fault diagnosis system described above is a powerful tool to aid an electronics technician in diagnosing faults in a digital circuit of nearly any size. It allows the technician not only to interactively observe the model and test the faulty unit, but also to allow the system to make suggestions for the location of the next best test. Although the discussion above has been limited to digital circuits, the same principles of model-based reasoning can be applied to analog circuits as well.

The power of this system lies in the fact that it utilizes knowledge about how circuits operate and applies it to the diagnosis of many different types of circuits, something which is inherently impossible in the use of ATEs and renders their use terribly inefficient. The potential gains

for the Navy in continuing research in this field are tremendous. It is conceivable that through the implementation of a generic fault diagnosis system in place of ATEs, the Navy could save millions of dollars and thousands of man hours and reduce equipment down-time while enhancing proper hardware operation.

5. ACKNOWLEDGMENTS

Special thanks are extended to Professor Pat Harrison, Computer Science Department U.S. Naval Academy, whose assistance and guidance made this Trident Research Project possible.

BIBLIOGRAPHY

Addis, T.R. *Designing Knowledge Based Systems*. (Prentice-Hall, Englewood Cliffs, N.J., 1986).

Author, *IEEE Guide to the Use of Atlas*. (IEEE, New York, 1984).

Author, *IEEE Standard ATLAS Test Language*. (IEEE, New York, 1984).

Ben-Basset, M. "Myopic Policies in Sequential Analysis." *IEEE Transactions on Computers*. 27(1978), 170-174.

Ben-Bassat, M. "Multimembership and Multiperspective Classification: Introduction, Applications, and a Bayesian Model." *IEEE Transactions On Systems, Man and Cybernetics*. 10(1980), 331-336.

Bobrow, D.G. (Ed.) *Qualitative Reasoning About Physical Systems*. (MIT press, Cambridge Mass., 1985).

de Kleer, J. "AI Approaches to Troubleshooting." in J.J. Richardson (Ed.) *Artificial Intelligence in Maintenance*. (Noyes Publications, Park Ridge, N.J., 1985) 79-89.

deKleer, J. "Causal and Teleological Reasoning In Circuit Recognition." Technical Report AI-TR-529, MIT, Cambridge, Mass., 1979.

de Kleer, J. & B.C. Williams. "Diagnosing Multiple Faults." *Artificial Intelligence*. 32(1987) 97-130.

Geffner, H. & J. Pearl. "Distributed Diagnosis of Systems With Multiple Faults." in: *Proceedings Third Conference on Artificial Intelligence Applications*. Orlando, Fl(1987) 224-230.

Hamscher, Walter. *Using Structural and Functional Information in Diagnostic Design*. Diss. Massachusetts Institute of Technology at Boston, 1979.

Marrone, M.P. & W.M. Spears. "FIS Users Guide" Rev 1.3., Unpublished Guide, 1987.

Moorthy, V.S. & B. Chandrasekaran. "A Representation For The Functioning of Devices That Support Compilation Of Expert Problem Solving Structures." in: J.J. Richardson (Ed.) *Artificial Intelligence in Maintenance*. (Noyes Publications, Park Ridge, N.J., 1985) 123-143.

Pazzani, Michael J. "Failure-Driven Learning of Fault Diagnosis Heuristics." *IEEE Transactions*. SMC-17 (1987) 380-394.

Pearl, J. "Fusion, Propagation, and Structuring in Belief Networks." *Artificial Intelligence*. 29(1986) 241-288.

Peng, Y. and James A. Reggia. "Plausibility of Diagnostic Hypotheses: The Nature of Simplicity." in: *Proceedings AAAI-86 I*. Philadelphia, PA (1986) 140-145.

Pipitone, F., K. DeJong, W. Spears, & M. Marrone. "The FIS Electronic Troubleshooting Project", Unpublished draft report, 1987.

Roylance, G. "Simple Models of Circuit Design." Technical Report 703, MIT, Cambridge, Mass., 1980.

Scarl, E.A., J.R. Jamieson & C.I. Delaune. "Diagnosis and Sensor Validation through Knowledge of Structure and Function." *IEEE Transactions*. SMC-17 (1987), 360-368.

Shore, J.E. "Relative Entropy, Probabilistic Inference and AI." in: Kanal, L.N. & Lemmer, J.F. (Eds.) *Uncertainty in Artificial Intelligence*. (Elsevier Science Publishers, North Holland, 1986), 211-215.

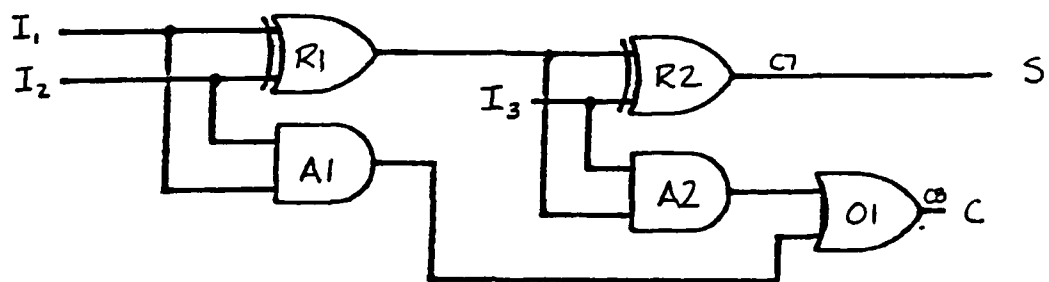


Figure 1. Simple Adder

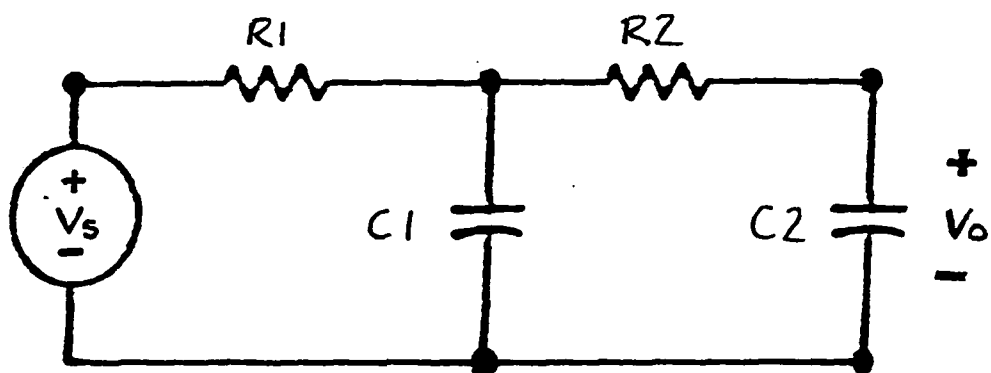


Figure 2. Cascaded Low-Pass Filter

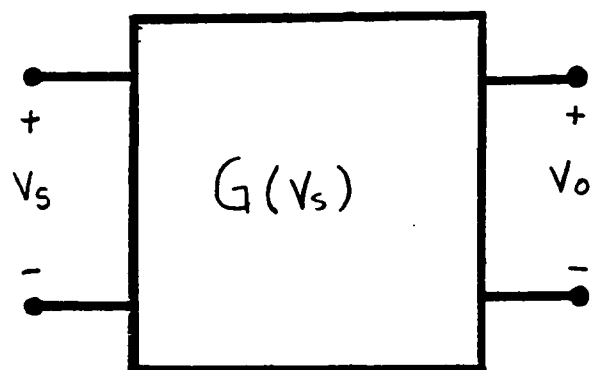


Figure 3. Black-Box Filter

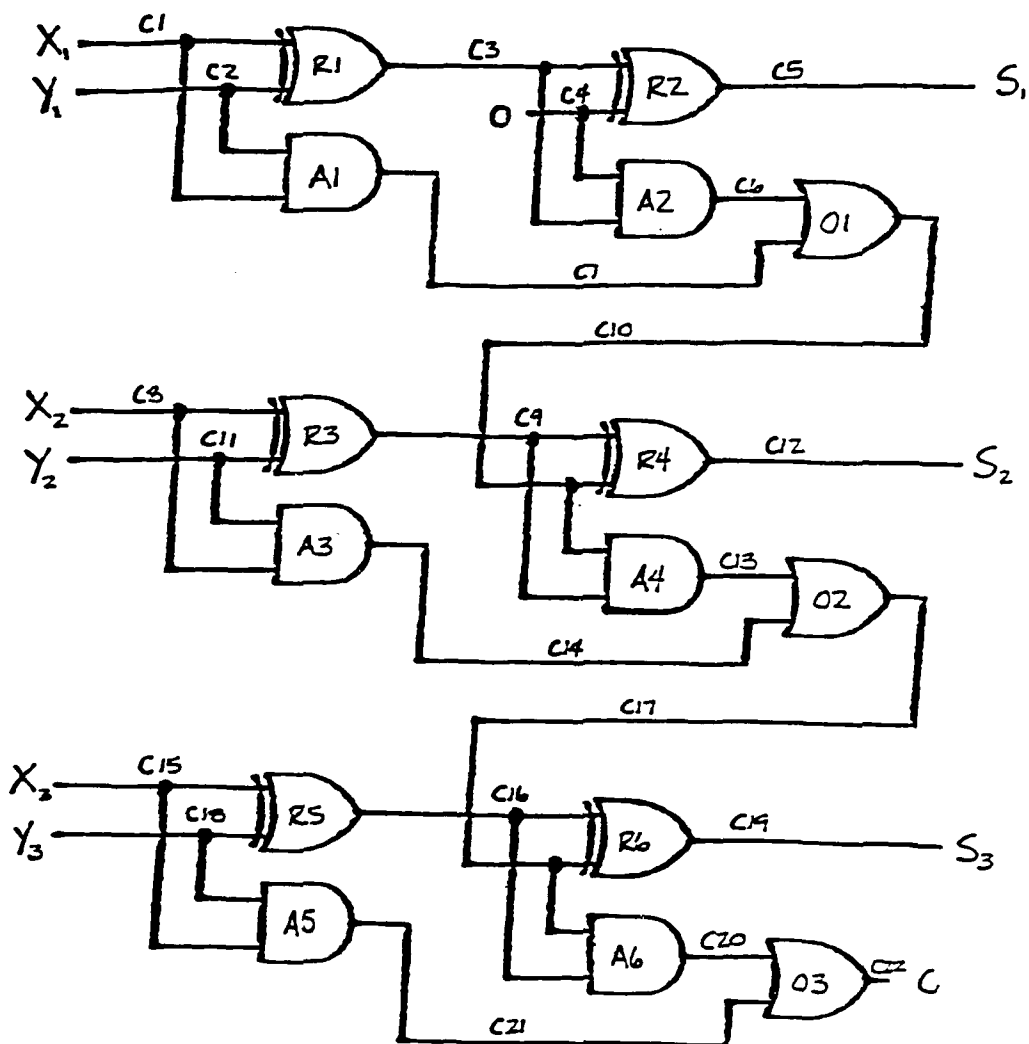


Figure 4. 3-Bit Adder

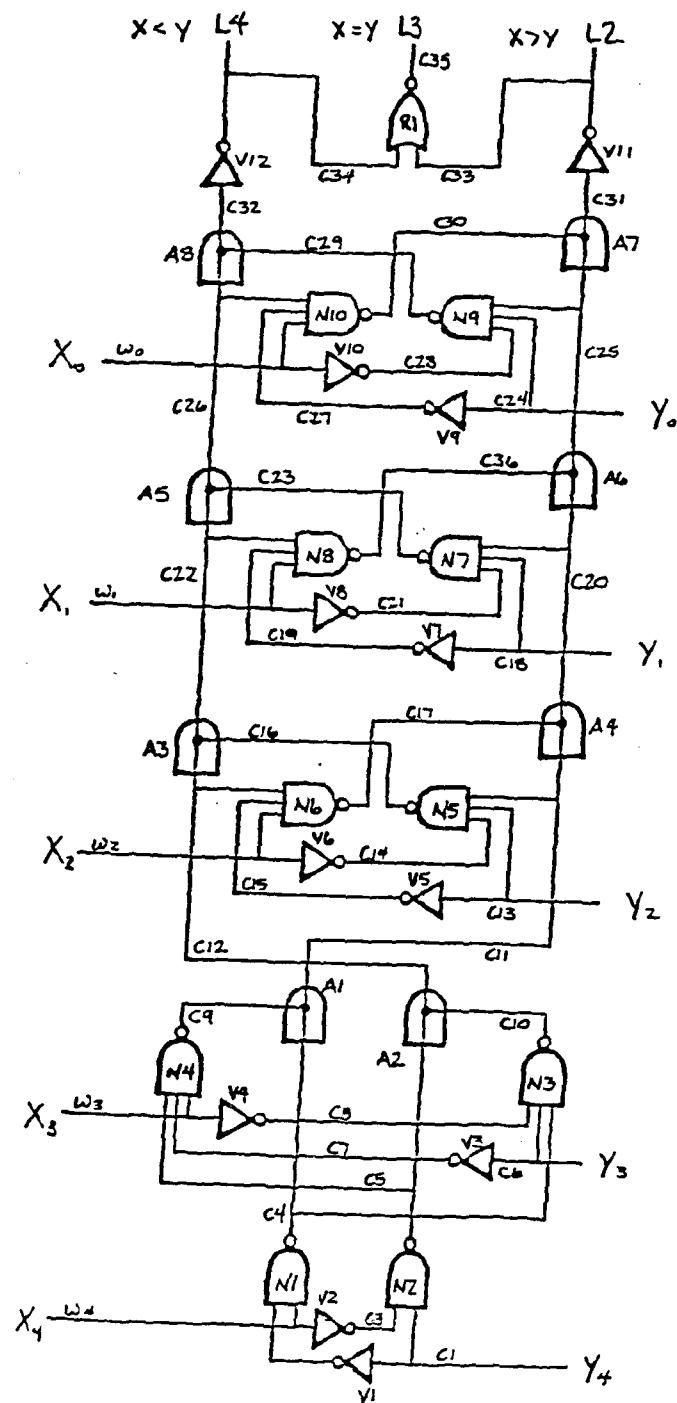


Figure 5. 5-Bit Comparator

APPENDIX A

```

;;; -*- Mode:LISP; Package:GW; -*-
;;;
;;; GoldWorks Knowledge Base
;;;
;;; Dumped on 12:43pm 31-Mar-88
;;; For registered user: HARRISON USNA
;;;
;;; Portions of Knowledge Base saved:
;;; (FRAME RELATION INSTANCE ASSERTION RULE)
;;;
(in-package 'gw)

```

```

(DEFINE-FRAME THE-DIAGNOSIS-SYSTEM
  (:print-name "THE-DIAGNOSIS-SYSTEM"
   :doc-string ""
   :is TOP-FRAME))

```

```

(DEFINE-FRAME CIRCUIT-MODULE
  (:print-name "CIRCUIT-MODULE"
   :doc-string ""
   :is TOP-FRAME)
  (SOURCE-LIST))

```

```

(DEFINE-FRAME INFERENCE-ENGINE
  (:print-name "INFERENCE-ENGINE"
   :doc-string ""
   :is THE-DIAGNOSIS-SYSTEM)
  (INPUT-NUM
   :default-values (0)
   :constraints (:LISP-TYPE NUMBER))
  (INPUT-LIST)
  (OUTPUT-LIST)
  (INPUT-PORT-LIST)
  (OUTPUT-PORT-LIST)
  (JUNCTION-LIST)
  (CONNECTOR-LIST)
  (ISET-NUM
   :default-values (0)
   :constraints (:LISP-TYPE NUMBER))
  (INPUT-SET-LIST)
  (OUTPUT-STATE
   :default-values (0)
   :constraints (:ONE-OF (0 1)))
  (ACTUAL-OUTPUT-LIST)
  (SUSPECT-LIST)
  (TEST-LIST)
  (TEST-STATE
   :default-values (0))

```

```

        :constraints (:ONE-OF (0 1)))
(TESTMADE-LIST)
(BEST-TEST)
(REMOVED-LIST)
(JUST-REMOVED)
(FAULT)
(FAULTY-LIST))

(DEFINE-FRAME USER-INTERFACE
  (:print-name "USER-INTERFACE"
   :doc-string ""
   :is THE-DIAGNOSIS-SYSTEM)
  (STATUS
   :constraints (:ONE-OF (:NO :VIEW :CLEAR :CLEAR-SINGLE
:CREATE-INPUTS :APPLY-INPUTS :BEST-TEST :MAKE-TEST
:CHECK-OUTPUT :FORGET-ALL :DIAGNOSE-NEXT)))
  (DISPLAY-MODULE)
  (INSTRUCTIONS))

(DEFINE-FRAME JUNCTION
  (:print-name "JUNCTION"
   :doc-string ""
   :is CIRCUIT-MODULE))

(DEFINE-FRAME BOARD-PORT
  (:print-name "BOARD-PORT"
   :doc-string ""
   :is CIRCUIT-MODULE)
  (VAL
   :constraints (:ONE-OF (0 1)))
  (CONN))

(DEFINE-FRAME CONNECTOR
  (:print-name "CONNECTOR"
   :doc-string ""
   :is CIRCUIT-MODULE)
  (VAL
   :constraints (:ONE-OF (0 1)))
  (TEST-REMOVED)
  (TEST-DEPENDENT))

(DEFINE-FRAME 2-1JUNCTION
  (:print-name "2-1JUNCTION"
   :doc-string ""
   :is JUNCTION)
  (I1-VAL
   :constraints (:ONE-OF (0 1)))
  (I2-VAL
   :constraints (:ONE-OF (0 1)))
  (O-VAL
   :constraints (:ONE-OF (0 1)))
  (I1-CONN))

```

```

(I2-CONN)
(O-CONN))

(DEFINE-FRAME 3-1JUNCTION
  (:print-name "3-1JUNCTION"
   :doc-string ""
   :is JUNCTION)
  (I1-CONN)
  (I2-CONN)
  (I3-CONN)
  (I1-VAL
   :when-modified (NAND-3I-TRANSFER-FUNCTION)
   :constraints (:ONE-OF (0 1)))
  (I2-VAL
   :when-modified (NAND-3I-TRANSFER-FUNCTION)
   :constraints (:ONE-OF (0 1)))
  (I3-VAL
   :when-modified (NAND-3I-TRANSFER-FUNCTION)
   :constraints (:ONE-OF (0 1)))
  (O-CONN)
  (O-VAL
   :constraints (:ONE-OF (0 1))))

(DEFINE-FRAME 1-1JUNCTION
  (:print-name "1-1JUNCTION"
   :doc-string ""
   :is JUNCTION)
  (I-CONN)
  (I-VAL
   :when-modified (INVERTER-TRANSFER-FUNCTION)
   :constraints (:ONE-OF (0 1)))
  (O-CONN)
  (O-VAL
   :constraints (:ONE-OF (0 1))))

(DEFINE-FRAME BOARD-OUTPUT-PORT
  (:print-name "BOARD-OUTPUT-PORT"
   :doc-string ""
   :is BOARD-PORT)
  (VAL
   :when-modified (UPDATE-OUTPUT-LIST)))

(DEFINE-FRAME BOARD-INPUT-PORT
  (:print-name "BOARD-INPUT-PORT"
   :doc-string ""
   :is BOARD-PORT)
  (VAL
   :when-modified (UPDATE-INPUT-LIST)))

(DEFINE-FRAME 2-PORT-CONNECTOR
  (:print-name "2-PORT-CONNECTOR"
   :doc-string ""

```

```

      :is CONNECTOR)
    (VAL)
    (CONN-1)
    (CONN-2))

(DEFINE-FRAME 3-PORT-CONNECTOR
  (:print-name "3-PORT-CONNECTOR"
   :doc-string ""
   :is CONNECTOR)
  (CONN-1)
  (CONN-2)
  (CONN-3))

(DEFINE-FRAME NANDGATE
  (:print-name "NANDGATE"
   :doc-string ""
   :is 2-1JUNCTION)
  (I2-VAL
   :when-modified (NAND-TRANSFER-FUNCTION))
  (I1-VAL
   :when-modified (NAND-TRANSFER-FUNCTION)))

(DEFINE-FRAME NORGATE
  (:print-name "NORGATE"
   :doc-string ""
   :is 2-1JUNCTION)
  (I2-VAL
   :when-modified (NOR-TRANSFER-FUNCTION))
  (I1-VAL
   :when-modified (NOR-TRANSFER-FUNCTION)))

(DEFINE-FRAME ORGATE
  (:print-name "ORGATE"
   :doc-string ""
   :is 2-1JUNCTION)
  (I2-VAL
   :when-modified (OR-TRANSFER-FUNCTION))
  (I1-VAL
   :when-modified (OR-TRANSFER-FUNCTION)))

(DEFINE-FRAME ANDGATE
  (:print-name "ANDGATE"
   :doc-string ""
   :is 2-1JUNCTION)
  (I2-VAL
   :when-modified (AND-TRANSFER-FUNCTION))
  (I1-VAL
   :when-modified (AND-TRANSFER-FUNCTION)))

(DEFINE-FRAME XORGATE
  (:print-name "XORGATE"
   :doc-string ""

```

```

      :is 2-1JUNCTION)
(I2-VAL
  :when-modified (XOR-TRANSFER-FUNCTION))
(I1-VAL
  :when-modified (XOR-TRANSFER-FUNCTION)))

(DEFINE-FRAME NANDGATE-3I
  (:print-name "NANDGATE-3I"
   :doc-string ""
   :is 3-1JUNCTION))

(DEFINE-FRAME INVERTER
  (:print-name "INVERTER"
   :doc-string ""
   :is 1-1JUNCTION))

(DEFINE-INSTANCE INF-ENG-DATA
  (:is INFERENCE-ENGINE))

(DEFINE-INSTANCE Y0
  (:print-name "Y0"
   :doc-string ""
   :is BOARD-INPUT-PORT)
  (SOURCE-LIST (Y0))
  (CONN C24)
  (VAL 0)
  )

(DEFINE-INSTANCE Y1
  (:print-name "Y1"
   :doc-string ""
   :is BOARD-INPUT-PORT)
  (SOURCE-LIST (Y1))
  (CONN C18)
  (VAL 0)
  )

(DEFINE-INSTANCE Y2
  (:print-name "Y2"
   :doc-string ""
   :is BOARD-INPUT-PORT)
  (SOURCE-LIST (Y2))
  (CONN C13)
  (VAL 1)
  )

(DEFINE-INSTANCE Y3
  (:print-name "Y3"
   :doc-string ""
   :is BOARD-INPUT-PORT)
  (SOURCE-LIST (Y3))
  (CONN C6)
  )

```

```

(VAL 0)
)

(DEFINE-INSTANCE Y4
  (:print-name "Y4"
   :doc-string ""
   :is BOARD-INPUT-PORT)
  (SOURCE-LIST (Y4))
  (CONN C1)
  (VAL 0)
)

(DEFINE-INSTANCE X1
  (:print-name "X1"
   :doc-string ""
   :is BOARD-INPUT-PORT)
  (SOURCE-LIST (X1))
  (CONN W1)
  (VAL 0)
)

(DEFINE-INSTANCE X0
  (:print-name "X0"
   :doc-string ""
   :is BOARD-INPUT-PORT)
  (SOURCE-LIST (X0))
  (CONN W0)
  (VAL 0)
)

(DEFINE-INSTANCE X2
  (:print-name "X2"
   :doc-string ""
   :is BOARD-INPUT-PORT)
  (SOURCE-LIST (X2))
  (CONN W2)
  (VAL 1)
)

(DEFINE-INSTANCE X3
  (:print-name "X3"
   :doc-string ""
   :is BOARD-INPUT-PORT)
  (SOURCE-LIST (X3))
  (CONN W3)
  (VAL 0)
)

(DEFINE-INSTANCE X4
  (:print-name "X4"
   :doc-string ""
   :is BOARD-INPUT-PORT)

```

```

(SOURCE-LIST (X4))
(CONN W4)
(VAL 0)
)

(DEFINE-INSTANCE L2
  (:print-name "L2"
   :doc-string ""
   :is BOARD-OUTPUT-PORT)
  (SOURCE-LIST
    (L2 C33 V11 C31 A7 C25 A6 C36 N8 C19 V7 C30 N10 W0
      X0 C26 A5 C22 A3 C16 N5 C14 V6 C23 N7 C18 Y1 C20 A4 C17
      N6 C12 A2 C10 N3 C8 V4 C15 V5 C13 Y2 W2 X2 C11 A1 C4 N1
      C2 V1 C9 N4 W3 X3 C5 N2 C1 Y4 C3 V2 W4 X4 C7 V3 C6 Y3
      C21 V8 W1 X1 C27 V9 C24 Y0))
  (CONN C33)
  (VAL 0)
  )

(DEFINE-INSTANCE L3
  (:print-name "L3"
   :doc-string ""
   :is BCARD-OUTPUT-PORT)
  (SOURCE-LIST
    (L3 C35 R1 C33 V11 C31 A7 C30 N10 C27 V9 C34 V12 C3
      A8 C26 A5 C23 N7 C21 V8 C29 N9 C25 A6 C20 A4 C17 C15 V5
      C36 N8 C22 A3 C12 A2 C10 N3 C8 V4 C16 N5 C11 A1 C4 N1
      C2 V1 C9 N4 W3 X3 C5 N2 C1 Y4 C3 V2 W4 X4 C7 V3 C6 Y3
      C13 Y2 C14 V6 W2 X2 C19 V7 C18 Y1 W1 X1 C28 V10 W0 X0
      C24 Y0))
  (CONN C35)
  (VAL 1)
  )

(DEFINE-INSTANCE L4
  (:print-name "L4"
   :doc-string ""
   :is BOARD-OUTPUT-PORT)
  (SOURCE-LIST
    (L4 C34 V12 C32 A8 C26 A5 C23 N7 C21 V8 C29 N9 C25
      A6 C20 A4 C17 N6 C15 V5 C36 N8 C22 A3 C12 A2 C10 N3 C8
      V4 C16 N5 C11 A1 C4 N1 C2 V1 C9 N4 W3 X3 C5 N2 C1 Y4 C3
      V2 W4 X4 C7 V3 C6 Y3 C13 Y2 C14 V6 W2 X2 C19 V7 C18 Y1
      W1 X1 C28 V10 W0 X0 C24 Y0))
  (CONN C34)
  (VAL 0)
  )

(DEFINE-INSTANCE DIAGNOSIS-FORM
  (:print-name "DIAGNOSIS-FORM"
   :doc-string ""
   :is SCREEN-TEMPLATE)

```



```

(OBJECTS
  ( (ACTUAL-OLIST :SLOT INF-ENG-DATA
      ACTUAL-OUTPUT-LIST :COLOR :BLUE)
    (SUSPECT-LIST :SLOT INF-ENG-DATA SUSPECT-LIST
      :COLOR :RED)
    (TEST-LIST :SLOT INF-ENG-DATA TEST-LIST)
    (JUST-REMOVED :SLOT INF-ENG-DATA JUST-REMOVED)
    (FAULT :SLOT INF-ENG-DATA FAULT :COLOR :RED)
    (FAULTY-LIST :SLOT INF-ENG-DATA FAULTY-LIST :COLOR
      :RED)
    (BEST-TEST :SLOT INF-ENG-DATA BEST-TEST :COLOR
      :GREEN)))
(CONTENTS
  ( ("Actual Outputs: " (:NO-SELECT ACTUAL-OLIST))
    ("Suspect List: " (:NO-SELECT SUSPECT-LIST))
    ("Test List: " (:NO-SELECT TEST-LIST))
    ("Just Removed: " (:NO-SELECT JUST-REMOVED))
    ("Fault: " (:NO-SELECT FAULT))
    ("Faulty List: " (:NO-SELECT FAULTY-LIST))
    ("Best Test: " (:NO-SELECT BEST-TEST))))
(TITLE "* Inference Engine *")
(TEXT-COLOR :LIGHT-GRAY)
(BORDER-COLOR :LIGHT-GRAY)
(BORDER :SINGLE)
(ORIENTATION :ROW)
(LAYOUT :LINEAR)
(SPACES-BETWEEN-COLUMNS 0)
(SPACES-BETWEEN-ROWS 0)
)

(DEFINE-INSTANCE WORK-SCREEN
  (:print-name "WORK-SCREEN"
    :doc-string "Main screen for circuit acquisition and
troubleshooting"
    :is SCREEN-LAYOUT)
  (MENU-BAR-ITEMS
    ( (" System "
      ( (:SLOT FAULT-DIAGNOSIS-SYSTEM NEW-SCREEN
        "End Diagnosis Session" WELCOME-SCREEN)))
      (" Circuit "
      ( (:MENU ACQUISITION-MENU
        "Enter Circuit Module and Connect")
        (:SLOT DIAGNOSIS-SESSION STATUS
        "Examine/Modify Circuit Module" :VIEW)
        (:SLOT DIAGNOSIS-SESSION STATUS
        "Clear all Module Instances" :CLEAR)
        (:SLOT DIAGNOSIS-SESSION STATUS
        "Clear Single Module Instance"
        :CLEAR-SINGLE)
        (:MENU TEST-MENU
        "Test Current Circuit for Correctness"))))
      (" Diagnosis "

```

```

( (:SLOT DIAGNOSIS-SESSION STATUS
  "Create an Input Set" :CREATE-INPUTS)
  (:SLOT DIAGNOSIS-SESSION STATUS
    "Apply Next Input Set" :APPLY-INPUTS)
  (:SLOT DIAGNOSIS-SESSION STATUS "Test Outputs"
    :CHECK-OUTPUT)
  (:SLOT DIAGNOSIS-SESSION STATUS "Best Test"
    :BEST-TEST)
  (:SLOT DIAGNOSIS-SESSION STATUS "Make Test"
    :MAKE-TEST)
  (:SLOT DIAGNOSIS-SESSION STATUS
    "Diagnose Next Fault" :DIAGNOSE-NEXT)
  (:SLOT DIAGNOSIS-SESSION STATUS "Forget All"
    :FORGET-ALL)))))
(MENU-BAR-BORDER-COLOR :LIGHT-GRAY)
(MENU-BAR-TEXT-COLOR :LIGHT-GRAY)
(SCREEN-TEMPLATES
  ( (CIRCUIT-FORM :LEFT 0 :TOP 11 :WIDTH 80 :HEIGHT 5)
    (DIAGNOSIS-FORM :LEFT 0 :TOP 16 :WIDTH 80 :HEIGHT
      9)))
(PARENT-SCREEN-CONTROL FAULT-DIAGNOSIS-SYSTEM)
)

(DEFINE-INSTANCE V12
  (:print-name "V12"
    :doc-string ""
    :is INVERTER)
  (SOURCE-LIST
    (V12 C32 A8 C26 A5 C23 N7 C21 V8 C29 N9 C25 A6 C20
      A4 C17 N6 C15 V5 C36 N8 C22 A3 C12 A2 C10 N3 C8 V4 C16
      N5 C11 A1 C4 N1 C2 V1 C9 N4 W3 X3 C5 N2 C1 Y4 C3 V2 W4
      X4 C7 V3 C6 Y3 C13 Y2 C14 V6 W2 X2 C19 V7 C18 Y1 W1 X1
      C28 V10 W0 X0 C24 Y0))
  (I-CONN C32)
  (I-VAL 1)
  (O-CONN C34)
  (O-VAL 0)
  )

(DEFINE-INSTANCE V11
  (:print-name "V11"
    :doc-string ""
    :is INVERTER)
  (SOURCE-LIST
    (V11 C31 A7 C25 A6 C36 N8 C19 V7 C30 N10 W0 X0 C26
      A5 C22 A3 C16 N5 C14 V6 C23 N7 C18 Y1 C20 A4 C17 N6 C12
      A2 C10 N3 C8 V4 C15 V5 C13 Y2 W2 X2 C11 A1 C4 N1 C2 V1
      C9 N4 W3 X3 C5 N2 C1 Y4 C3 V2 W4 X4 C7 V3 C6 Y3 C21 V8
      W1 X1 C27 V9 C24 Y0))
  (I-CONN C31)
  (I-VAL 1)
  (O-CONN C33)
  )

```

```

(O-VAL 0)
)

(DEFINE-INSTANCE MAKE-TEST-MENU
  (:print-name "MAKE-TEST-MENU"
   :doc-string ""
   :is SET-SLOT-VALUES)
  (INSTRUCTIONS ("Test Node: " C15))
  (BORDER-COLOR :BLUE)
  (TOP 2)
  (LEFT 0)
  (CENTER :NO-CENTERING)
  (REVERT-BUTTON :NO)
  (DEFAULT-BUTTON :NO)
  (CONTENTS
    ( (INF-ENG-DATA TEST-STATE
       "Actual Value at Test Point: ")))
  )

(DEFINE-INSTANCE FORGET-CONFIRM-MENU
  (:print-name "FORGET-CONFIRM-MENU"
   :doc-string ""
   :is POPUP-CONFIRM)
  (BORDER-COLOR :RED)
  (CENTER :X-AND-Y)
  (ANSWER :YES)
  (DEFAULT-ANSWER :NO)
  (CONTENTS
    (:RETURN " Forget all diagnosis " :RETURN
     " done on this circuit? "))
  )

(DEFINE-INSTANCE CHECK-OUTPUT-MENU
  (:print-name "CHECK-OUTPUT-MENU"
   :doc-string ""
   :is SET-SLOT-VALUES)
  (INSTRUCTIONS
    ("Output Port: " L4 :RETURN "With Inputs: "
     ( (Y4 0) (Y3 0) (Y2 1) (Y1 0) (Y0 0)
       (X4 0) (X3 0) (X2 1) (X1 0) (X0 0))))
  (BORDER-COLOR :BLUE)
  (TOP 2)
  (LEFT 0)
  (CENTER :NO-CENTERING)
  (REVERT-BUTTON :NO)
  (DEFAULT-BUTTON :NO)
  (CONTENTS
    ( (INF-ENG-DATA OUTPUT-STATE
       "Actual Value at Output Port: ")))
  )

```

```

(DEFINE-INSTANCE DISPLAY-MODULE-MENU
  (:print-name "DISPLAY-MODULE-MENU"
   :doc-string ""
   :is SET-SLOT-VALUES)
  (INSTRUCTIONS ("Circuit Module: " X0))
  (BORDER-COLOR :BLUE)
  (TOP 2)
  (LEFT 0)
  (CENTER :NO-CENTERING)
  (REVERT-BUTTON :NO)
  (DEFAULT-BUTTON :NO)
  (CONTENTS
    ( (X0 VAL "Value at Port: ")
      (X0 CONN "Connected to: ")))
  )

(DEFINE-INSTANCE DIAGNOSIS-SESSION
  (:print-name "DIAGNOSIS-SESSION"
   :doc-string ""
   :is USER-INTERFACE)
  (STATUS :NO)
  (INSTRUCTIONS ("Circuit Module: " X0))
  )

(DEFINE-INSTANCE ACQUISITION-MENU
  (:print-name "ACQUISITION-MENU"
   :doc-string ""
   :is POPUP-CHOOSE)
  (TITLE "Circuit Module Type")
  (BORDER-COLOR :BLUE)
  (CENTER :X-AND-Y)
  (CONTENTS
    ( (" Board Port " (QUOTE BOARD-PORT))
      (" Connector " (QUOTE CONNECTOR))
      (" 1-1 Junction " (QUOTE 1-1JUNCTION))
      (" 2-1 Junction " (QUOTE 2-1JUNCTION))
      (" 3-1 Junction " (QUOTE 3-1JUNCTION)))
  )

(DEFINE-INSTANCE FAULT-DIAGNOSIS-SYSTEM
  (:print-name "FAULT-DIAGNOSIS-SYSTEM"
   :doc-string "Fault Diagnosis System is a system to
troubleshoot circuits"
   :is SCREEN-CONTROL)
  (SCREEN-LAYOUTS (WELCOME-SCREEN WORK-SCREEN))
  (TEMPLATE-AREA-HEIGHT 23)
  (NEW-SCREEN :NO)
  (COUNT 0)
  (STATUS :INITIAL-STATUS)
  (CLOSE-REQUEST :YES)
  )

```

```

(DEFINE-INSTANCE WELCOME-SCREEN
  (:print-name "WELCOME-SCREEN"
   :doc-string "Initial screen for fault diagnosis system"
   :is SCREEN-LAYOUT)
  (MENU-BAR-ITEMS
   ( (" Start/Quit "
      ( (:SLOT FAULT-DIAGNOSIS-SYSTEM NEW-SCREEN
          "Begin Diagnosis Session" WORK-SCREEN)
        (:SLOT FAULT-DIAGNOSIS-SYSTEM CLOSE-REQUEST
          "Exit MULTFAULT" :YES))))))
  (MENU-BAR-BORDER-COLOR :LIGHT-GRAY)
  (MENU-BAR-TEXT-COLOR :LIGHT-GRAY)
  (SCREEN-TEMPLATES
   ( (WELCOME-FORM :LEFT 18 :TOP 5 :WIDTH 40
      :HEIGHT 10)))
  (PARENT-SCREEN-CONTROL FAULT-DIAGNOSIS-SYSTEM)
  )

(DEFINE-INSTANCE WELCOME-FORM
  (:print-name "WELCOME-FORM"
   :doc-string "Form to display on the initial screen"
   :is SCREEN-TEMPLATE)
  (CONTENTS
   ( ("          MULTFAULT") ("")
     ("    The circuit troubleshooting")
     ("          expert system") ("")
     ("    Midshipman First Class ")
     ("    Bryan P. Graham") ("")))
  (TEXT-COLOR :LIGHT-GRAY)
  (BORDER-COLOR :LIGHT-GRAY)
  (BORDER :DOUBLE)
  (ORIENTATION :ROW)
  (LAYOUT :LINEAR)
  (SPACES-BETWEEN-COLUMNS 0)
  (SPACES-BETWEEN-ROWS 0)
  )

(DEFINE-INSTANCE MODULE-NAME-MENU
  (:print-name "MODULE-NAME-MENU"
   :doc-string ""
   :is POPUP-ASK-USER)
  (INSTRUCTIONS "Circuit Module Name")
  (BORDER-COLOR :BLUE)
  (CENTER :X-AND-Y)
  (ANSWER V12)
  (LAST-ANSWER :INITIAL-STATUS)
  (ANSWER-WIDTH 10)
  (CONTENTS
   ("The Name of the " MOD-TYPE " component is:"))
  )

```

```

(DEFINE-INSTANCE 2-1JUNCTION-MENU
  (:print-name "2-1JUNCTION-MENU"
   :doc-string ""
   :is POPUP-CHOOSE)
  (INSTRUCTIONS "Which Type of Junction?")
  (BORDER-COLOR :BLUE)
  (CENTER :X-AND-Y)
  (ANSWER (QUOTE NORGATE))
  (CONTENTS
    ( ("AND Gate" (QUOTE ANDGATE))
      ("NAND Gate" (QUOTE NANDGATE))
      ("OR Gate" (QUOTE ORGATE))
      ("NOR Gate" (QUOTE NORGATE))
      ("XOR Gate" (QUOTE XORGATE)))))
  (FORCE-CHOICE :YES)
  )

(DEFINE-INSTANCE MODULE-CONN-MENU
  (:print-name "MODULE-CONN-MENU"
   :doc-string ""
   :is POPUP-ASK-USER)
  (INSTRUCTIONS "Output Port:")
  (BORDER-COLOR :BLUE)
  (CENTER :X-AND-Y)
  (ANSWER C34)
  (LAST-ANSWER :INITIAL-STATUS)
  (ANSWER-WIDTH 10)
  (CONTENTS (MOD-TYPE " " MOD-NAME " is connected to:"))
  )

(DEFINE-INSTANCE CONNECTOR-MENU
  (:print-name "CONNECTOR-MENU"
   :doc-string ""
   :is POPUP-CHOOSE)
  (INSTRUCTIONS "Which Type of Connector?")
  (BORDER-COLOR :BLUE)
  (CENTER :X-AND-Y)
  (ANSWER (QUOTE 3-PORT-CONNECTOR))
  (CONTENTS
    ( ("2 Port Connector" (QUOTE 2-PORT-CONNECTOR))
      ("3 Port Connector" (QUOTE 3-PORT-CONNECTOR)))))
  (FORCE-CHOICE :YES)
  )

(DEFINE-INSTANCE BOARD-PORT-MENU
  (:print-name "BOARD-PORT-MENU"
   :doc-string ""
   :is POPUP-CHOOSE)
  (INSTRUCTIONS "Which Type of Board Port?")
  (BORDER-COLOR :BLUE)
  (CENTER :X-AND-Y)
  (ANSWER (QUOTE BOARD-INPUT-PORT))

```

```

(CONTENTS
  ( ("Input Port" (QUOTE BOARD-INPUT-PORT))
    ("Output Port" (QUOTE BOARD-OUTPUT-PORT)))
(FORCE-CHOICE :YES)
)

(DEFINE-INSTANCE NO-MODULE-MENU
  (:print-name "NO-MODULE-MENU"
   :doc-string ""
   :is POPUP-CHOOSE)
  (INSTRUCTIONS "Ooops!!")
  (BORDER-COLOR :BLUE)
  (TOP 2)
  (LEFT 60)
  (CENTER :NO-CENTERING)
  (ANSWER :NO)
  (TARGET-INSTANCE DIAGNOSIS-SESSION)
  (TARGET-SLOT STATUS)
  (CONTENTS
    ( (:TEXT "There are no") (:TEXT "circuit modules")
      ("OK" :NO)))
  )

(DEFINE-INSTANCE CLEAR-MODULE-CONFIRM-MENU
  (:print-name "CLEAR-MODULE-CONFIRM-MENU"
   :doc-string ""
   :is POPUP-CONFIRM)
  (BORDER-COLOR :RED)
  (CENTER :X-AND-Y)
  (ANSWER :NO)
  (DEFAULT-ANSWER :NO)
  (CONTENTS
    (:RETURN :RETURN "  DELETE all module instances??
  ))
  )

(DEFINE-INSTANCE CIRCUIT-FORM
  (:print-name "CIRCUIT-FORM"
   :doc-string ""
   :is SCREEN-TEMPLATE)
  (OBJECTS
    ( (INPUTNUM :SLOT INF-ENG-DATA INPUT-NUM :WIDTH 5
       :COLOR :WHITE)
      (ISET :SLOT INF-ENG-DATA ISET-NUM :WIDTH 5 :COLOR
       :WHITE)
      (INPUTS :SLOT INF-ENG-DATA INPUT-LIST :WIDTH 25
       :COLOR :WHITE)
      (OUTPUTS :SLOT INF-ENG-DATA OUTPUT-LIST :WIDTH 25
       :COLOR :WHITE)))
  (CONTENTS
    ( ("Input Set #" (:NO-SELECT INPUTNUM) " of "
      (:NO-SELECT ISET))
  )

```

```

        ("Inputs:" (:NO-SELECT INPUTS))
        ("Outputs:" (:NO-SELECT OUTPUTS)))
(TITLE "* Circuit *")
(TEXT-COLOR :LIGHT-GRAY)
(BORDER-COLOR :LIGHT-GRAY)
(BORDER :SINGLE)
(ORIENTATION :ROW)
(LAYOUT :LINEAR)
(SPACES-BETWEEN-COLUMNS 0)
(SPACES-BETWEEN-ROWS 0)
)

(DEFINE-INSTANCE MODULE-LIST-MENU
  (:print-name "MODULE-LIST-MENU"
   :doc-string ""
   :is POPUP-CHOOSE)
  (INSTRUCTIONS "Make Test at:")
  (BORDER-COLOR :BLUE)
  (TOP 2)
  (LEFT 60)
  (CENTER :NO-CENTERING)
  (ANSWER C15)
  (CONTENTS
    ( ("C33" C33) ("C31" C31) ("C31" C31) ("C25" C25)
      ("C30" C30) ("C30" C30) ("W0" W0) ("C27" C27)
      ("C26" C26) ("C27" C27) ("C24" C24) ("C34" C34)
      ("C32" C32) ("C32" C32) ("C26" C26) ("C29" C29)
      ("C26" C26) ("C22" C22) ("C23" C23) ("C23" C23)
      ("C18" C18) ("C20" C20) ("C21" C21) ("C21" C21)
      ("W1" W1) ("C29" C29) ("C24" C24) ("C25" C25)
      ("C28" C28) ("C25" C25) ("C20" C20) ("C36" C36)
      ("C20" C20) ("C11" C11) ("C17" C17) ("C17" C17)
      ("W2" W2) ("C15" C15) ("C12" C12) ("C15" C15)
      ("C13" C13) ("C36" C36) ("W1" W1) ("C19" C19)
      ("C22" C22) ("C22" C22) ("C12" C12) ("C16" C16)
      ("C16" C16) ("C11" C11) ("C13" C13) ("C14" C14)
      ("C14" C14) ("W2" W2) ("C19" C19) ("C18" C18)
      ("C28" C28) ("W0" W0)))
  )

(DEFINE-INSTANCE 1-1JUNCTION-MENU
  (:print-name "1-1JUNCTION-MENU"
   :doc-string ""
   :is POPUP-CHOOSE)
  (INSTRUCTIONS "Which Type of Junction?")
  (BORDER-COLOR :BLUE)
  (CENTER :X-AND-Y)
  (ANSWER (QUOTE INVERTER))
  (CONTENTS ( ("Inverter" (QUOTE INVERTER))))
  (FORCE-CHOICE :YES)
  )

```



```

      (:print-name "V10"
       :doc-string ""
       :is INVERTER)
(SOURCE-LIST (V10 W0 X0))
(I-CONN W0)
(I-VAL 0)
(O-CONN C28)
(O-VAL 1)
)

(DEFINE-INSTANCE V9
  (:print-name "V9"
   :doc-string ""
   :is INVERTER)
  (SOURCE-LIST (V9 C24 Y0))
  (I-CONN C24)
  (I-VAL 0)
  (O-CONN C27)
  (O-VAL 1)
)

(DEFINE-INSTANCE V8
  (:print-name "V8"
   :doc-string ""
   :is INVERTER)
  (SOURCE-LIST (V8 W1 X1))
  (I-CONN W1)
  (I-VAL 0)
  (O-CONN C21)
  (O-VAL 1)
)

(DEFINE-INSTANCE V7
  (:print-name "V7"
   :doc-string ""
   :is INVERTER)
  (SOURCE-LIST (V7 C18 Y1))
  (I-CONN C18)
  (I-VAL 0)
  (O-CONN C19)
  (O-VAL 1)
)

(DEFINE-INSTANCE V6
  (:print-name "V6"
   :doc-string ""
   :is INVERTER)
  (SOURCE-LIST (V6 W2 X2))
  (I-CONN W2)
  (I-VAL 1)
  (O-CONN C14)
  (O-VAL 0)
)

```

```

)

(DEFINE-INSTANCE V5
  (:print-name "V5"
   :doc-string ""
   :is INVERTER)
  (SOURCE-LIST (V5 C13 Y2))
  (I-CONN C13)
  (I-VAL 1)
  (O-CONN C15)
  (O-VAL 0)
)

(DEFINE-INSTANCE V4
  (:print-name "V4"
   :doc-string ""
   :is INVERTER)
  (SOURCE-LIST (V4 W3 X3))
  (I-CONN W3)
  (I-VAL 0)
  (O-CONN C8)
  (O-VAL 1)
)

(DEFINE-INSTANCE V3
  (:print-name "V3"
   :doc-string ""
   :is INVERTER)
  (SOURCE-LIST (V3 C6 Y3))
  (I-CONN C6)
  (I-VAL 0)
  (O-CONN C7)
  (O-VAL 1)
)

(DEFINE-INSTANCE V2
  (:print-name "V2"
   :doc-string ""
   :is INVERTER)
  (SOURCE-LIST (V2 W4 X4))
  (I-CONN W4)
  (I-VAL 0)
  (O-CONN C3)
  (O-VAL 1)
)

(DEFINE-INSTANCE V1
  (:print-name "V1"
   :doc-string ""
   :is INVERTER)
  (SOURCE-LIST (V1 C1 Y4))
  (I-CONN C1)
)

```

```

(I-VAL 0)
(O-CONN C2)
(O-VAL 1)
)

(DEFINE-INSTANCE C2
  (:print-name "C2"
   :doc-string ""
   :is 2-PORT-CONNECTOR)
  (SOURCE-LIST (C2 V1 C1 Y4))
  (VAL 1)
  (CONN-1 V1)
  (CONN-2 N1)
)

(DEFINE-INSTANCE C3
  (:print-name "C3"
   :doc-string ""
   :is 2-PORT-CONNECTOR)
  (SOURCE-LIST (C3 V2 W4 X4))
  (VAL 1)
  (CONN-1 V2)
  (CONN-2 N2)
)

(DEFINE-INSTANCE C7
  (:print-name "C7"
   :doc-string ""
   :is 2-PORT-CONNECTOR)
  (SOURCE-LIST (C7 V3 C6 Y3))
  (VAL 1)
  (CONN-1 V3)
  (CONN-2 N4)
)

(DEFINE-INSTANCE C8
  (:print-name "C8"
   :doc-string ""
   :is 2-PORT-CONNECTOR)
  (SOURCE-LIST (C8 V4 W3 X3))
  (VAL 1)
  (CONN-1 V4)
  (CONN-2 N3)
)

(DEFINE-INSTANCE C9
  (:print-name "C9"
   :doc-string ""
   :is 2-PORT-CONNECTOR)
  (SOURCE-LIST
    (C9 N4 W3 X3 C5 N2 C1 Y4 C3 V2 W4 X4 C7 V3 C6 Y3))
  (VAL 1)

```

```

(CONN-1 N4)
(CONN-2 A1)
)

(DEFINE-INSTANCE C10
  (:print-name "C10"
   :doc-string ""
   :is 2-PORT-CONNECTOR)
  (SOURCE-LIST
    (C10 N3 C8 V4 W3 X3 C6 Y3 C4 N1 W4 X4 C2 V1 C1 Y4))
  (VAL 1)
  (CONN-1 N3)
  (CONN-2 A2)
)

(DEFINE-INSTANCE C14
  (:print-name "C14"
   :doc-string ""
   :is 2-PORT-CONNECTOR)
  (SOURCE-LIST (C14 V6 W2 X2))
  (VAL 0)
  (CONN-1 V6)
  (CONN-2 N5)
)

(DEFINE-INSTANCE C15
  (:print-name "C15"
   :doc-string ""
   :is 2-PORT-CONNECTOR)
  (SOURCE-LIST (C15 V5 C13 Y2))
  (VAL 0)
  (CONN-1 V5)
  (CONN-2 N6)
)

(DEFINE-INSTANCE C16
  (:print-name "C16"
   :doc-string ""
   :is 2-PORT-CONNECTOR)
  (SOURCE-LIST
    (C16 N5 C11 A1 C4 N1 C2 V1 C9 N4 W3 X3 C5 N2 C1 Y4
     C3 V2 W4 X4 C7 V3 C6 Y3 C13 Y2 C14 V6 W2 X2))
  (VAL 1)
  (CONN-1 N5)
  (CONN-2 A3)
)

(DEFINE-INSTANCE C17
  (:print-name "C17"
   :doc-string ""

```

```

      :is 2-PORT-CONNECTOR)
(SOURCE-LIST
  (C17 N6 C12 A2 C10 N3 C8 V4 W3 X3 C6 Y3 C4 N1 C2 V1
   C5 N2 C1 Y4 C3 V2 W4 X4 C15 V5 C13 Y2 W2 X2))
(VAL 1)
(CONN-1 N6)
(CONN-2 A4)
)

(DEFINE-INSTANCE C19
  (:print-name "C19"
   :doc-string ""
   :is 2-PORT-CONNECTOR)
  (SOURCE-LIST (C19 V7 C18 Y1))
  (VAL 1)
  (CONN-1 V7)
  (CONN-2 N8)
  )

(DEFINE-INSTANCE C21
  (:print-name "C21"
   :doc-string ""
   :is 2-PORT-CONNECTOR)
  (SOURCE-LIST (C21 V8 W1 X1))
  (VAL 1)
  (CONN-1 V8)
  (CONN-2 N7)
  )

(DEFINE-INSTANCE C23
  (:print-name "C23"
   :doc-string ""
   :is 2-PORT-CONNECTOR)
  (SOURCE-LIST
    (C23 N7 C18 Y1 C20 A4 C17 N6 C12 A2 C10 N3 C8 V4 C15
     V5 C13 Y2 W2 X2 C11 A1 C4 N1 C2 V1 C9 N4 W3 X3 C5 N2 C1
     Y4 C3 V2 W4 X4 C7 V3 C6 Y3 C21 V8 W1 X1))
  (VAL 1)
  (CONN-1 N7)
  (CONN-2 A5)
  )

(DEFINE-INSTANCE C27
  (:print-name "C27"
   :doc-string ""
   :is 2-PORT-CONNECTOR)
  (SOURCE-LIST (C27 V9 C24 Y0))
  (VAL 1)
  (CONN-1 V9)
  (CONN-2 N10)
  )

```

```

(DEFINE-INSTANCE C28
  (:print-name "C28"
   :doc-string ""
   :is 2-PORT-CONNECTOR)
  (SOURCE-LIST (C28 V10 W0 X0))
  (VAL 1)
  (CONN-1 V10)
  (CONN-2 N9)
  )

```

```

(DEFINE-INSTANCE C29
  (:print-name "C29"
   :doc-string ""
   :is 2-PORT-CONNECTOR)
  (SOURCE-LIST
    (C29 N9 C25 A6 C20 A4 C17 N6 C15 V5 C36 N8 C22 A3
     C12 A2 C10 N3 C8 V4 C16 N5 C11 A1 C4 N1 C2 V1 C9 N4 W3
     X3 C5 N2 C1 Y4 C3 V2 W4 X4 C7 V3 C6 Y3 C13 Y2 C14 V6 W2
     X2 C19 V7 C18 Y1 W1 X1 C28 V10 W0 X0 C24 Y0))
  (VAL 1)
  (CONN-1 N9)
  (CONN-2 A8)
  )

```

```

(DEFINE-INSTANCE C30
  (:print-name "C30"
   :doc-string ""
   :is 2-PORT-CONNECTOR)
  (SOURCE-LIST
    (C30 N10 W0 X0 C26 A5 C22 A3 C16 N5 C14 V6 C23 N7
     C18 Y1 C20 A4 C17 N6 C12 A2 C10 N3 C8 V4 C15 V5 C13 Y2
     W2 X2 C11 A1 C4 N1 C2 V1 C9 N4 W3 X3 C5 N2 C1 Y4 C3 V2
     W4 X4 C7 V3 C6 Y3 C21 V8 W1 X1 C27 V9 C24 Y0))
  (VAL 1)
  (CONN-1 N10)
  (CONN-2 A7)
  )

```

```

(DEFINE-INSTANCE C31
  (:print-name "C31"
   :doc-string ""
   :is 2-PORT-CONNECTOR)
  (SOURCE-LIST
    (C31 A7 C25 A6 C36 N8 C19 V7 C30 N10 W0 X0 C26 A5
     C22 A3 C16 N5 C14 V6 C23 N7 C18 Y1 C20 A4 C17 N6 C12 A2
     C10 N3 C8 V4 C15 V5 C13 Y2 W2 X2 C11 A1 C4 N1 C2 V1 C9
     N4 W3 X3 C5 N2 C1 Y4 C3 V2 W4 X4 C7 V3 C6 Y3 C21 V8 W1
     X1 C27 V9 C24 Y0))
  (VAL 1)
  (CONN-1 A7)
  )

```



```

(CONN-2 V11)
)

(DEFINE-INSTANCE C32
  (:print-name "C32"
   :doc-string ""
   :is 2-PORT-CONNECTOR)
  (SOURCE-LIST
    (C32 A8 C26 A5 C23 N7 C21 V8 C29 N9 C25 A6 C20 A4
     C17 N6 C15 V5 C36 N8 C22 A3 C12 A2 C10 N3 C8 V4 C16 N5
     C11 A1 C4 N1 C2 V1 C9 N4 W3 X3 C5 N2 C1 Y4 C3 V2 W4 X4
     C7 V3 C6 Y3 C13 Y2 C14 V6 W2 X2 C19 V7 C18 Y1 W1 X1 C28
     C10 W0 X0 C24 Y0))
  (VAL 1)
  (CONN-1 A8)
  (CONN-2 V12)
  )

(DEFINE-INSTANCE C36
  (:print-name "C36"
   :doc-string ""
   :is 2-PORT-CONNECTOR)
  (SOURCE-LIST
    (C36 N8 C22 A3 C12 A2 C10 N3 C8 V4 C16 N5 C11 A1 C4
     N1 C2 V1 C9 N4 W3 X3 C5 N2 C1 Y4 C3 V2 W4 X4 C7 V3 C6
     Y3 C13 Y2 C14 V6 W2 X2 C19 V7 C18 Y1 W1 X1))
  (VAL 1)
  (CONN-1 N8)
  (CONN-2 A6)
  )

(DEFINE-INSTANCE C35
  (:print-name "C35"
   :doc-string ""
   :is 2-PORT-CONNECTOR)
  (SOURCE-LIST
    (C35 R1 C33 V11 C31 A7 C30 N10 C27 V9 C34 V12 C32 A8
     C26 A5 C23 N7 C21 V8 C29 N9 C25 A6 C20 A4 C17 N6 C15 V5
     C36 N8 C22 A3 C12 A2 C10 N3 C8 V4 C16 N5 C11 A1 C4 N1
     C2 V1 C9 N4 W3 X3 C5 N2 C1 Y4 C3 V2 W4 X4 C7 V3 C6 Y3
     C13 Y2 C14 V6 W2 X2 C19 V7 C18 Y1 W1 X1 C28 V10 W0 X0
     C24 Y0))
  (VAL 1)
  (CONN-1 R1)
  (CONN-2 L3)
  )

(DEFINE-INSTANCE C1
  (:print-name "C1"
   :doc-string ""
   :is 3-PORT-CONNECTOR)
  (SOURCE-LIST (C1 Y4))

```

```

(VAL 0)
(CONN-1 Y4)
(CONN-2 V1)
(CONN-3 N2)
)

(DEFINE-INSTANCE C4
  (:print-name "C4"
   :doc-string ""
   :is 3-PORT-CONNECTOR)
  (SOURCE-LIST (C4 N1 W4 X4 C2 V1 C1 Y4))
  (VAL 1)
  (CONN-1 N1)
  (CONN-2 N3)
  (CONN-3 A1)
  )

(DEFINE-INSTANCE C5
  (:print-name "C5"
   :doc-string ""
   :is 3-PORT-CONNECTOR)
  (SOURCE-LIST (C5 N2 C1 Y4 C3 V2 W4 X4))
  (VAL 1)
  (CONN-1 N2)
  (CONN-2 A2)
  (CONN-3 N4)
  )

(DEFINE-INSTANCE C6
  (:print-name "C6"
   :doc-string ""
   :is 3-PORT-CONNECTOR)
  (SOURCE-LIST (C6 Y3))
  (VAL 0)
  (CONN-1 Y3)
  (CONN-2 V3)
  (CONN-3 N3)
  )

(DEFINE-INSTANCE C11
  (:print-name "C11"
   :doc-string ""
   :is 3-PORT-CONNECTOR)
  (SOURCE-LIST
    (C11 A1 C4 N1 C2 V1 C9 N4 W3 X3 C5 N2 C1 Y4 C3 V2 W4
     X4 C7 V3 C6 Y3))
  (VAL 1)
  (TEST-REMOVED NIL)
  (TEST-DEPENDENT ( (C11 OK) (C12 OK)))
  (CONN-1 A1)
  (CONN-2 A4)
  (CONN-3 N5)
  )

```

```

)

(DEFINE-INSTANCE C12
  (:print-name "C12"
   :doc-string ""
   :is 3-PORT-CONNECTOR)
  (SOURCE-LIST
    (C12 A2 C10 N3 C8 V4 W3 X3 C6 Y3 C4 N1 C2 V1 C5 N2
     C1 Y4 C3 V2 W4 X4))
  (VAL 1)
  (TEST-REMOVED (V2 N2 V1 N1 V4 N3 A2))
  (TEST-DEPENDENT ( (C11 OK) (C12 OK)))
  (CONN-1 A2)
  (CONN-2 A3)
  (CONN-3 N6)
)

(DEFINE-INSTANCE C13
  (:print-name "C13"
   :doc-string ""
   :is 3-PORT-CONNECTOR)
  (SOURCE-LIST (C13 Y2))
  (VAL 1)
  (CONN-1 Y2)
  (CONN-2 V5)
  (CONN-3 N5)
)

(DEFINE-INSTANCE C18
  (:print-name "C18"
   :doc-string ""
   :is 3-PORT-CONNECTOR)
  (SOURCE-LIST (C18 Y1))
  (VAL 0)
  (CONN-1 Y1)
  (CONN-2 V7)
  (CONN-3 N7)
)

(DEFINE-INSTANCE C20
  (:print-name "C20"
   :doc-string ""
   :is 3-PORT-CONNECTOR)
  (SOURCE-LIST
    (C20 A4 C17 N6 C12 A2 C10 N3 C8 V4 C15 V5 C13 Y2 W2
     X2 C11 A1 C4 N1 C2 V1 C9 N4 W3 X3 C5 N2 C1 Y4 C3 V2 W4
     X4 C7 V3 C6 Y3))
  (VAL 1)
  (CONN-1 A4)
  (CONN-2 A6)
  (CONN-3 N7)
)

```

```

(DEFINE-INSTANCE C22
  (:print-name "C22"
   :doc-string ""
   :is 3-PORT-CONNECTOR)
  (SOURCE-LIST
    (C22 A3 C12 A2 C10 N3 C8 V4 C16 N5 C11 A1 C4 N1 C2
     V1 C9 N4 W3 X3 C5 N2 C1 Y4 C3 V2 W4 X4 C7 V3 C6 Y3 C13
     Y2 C14 V6 W2 X2))
  (VAL 1)
  (CONN-1 A3)
  (CONN-2 A5)
  (CONN-3 N8)
  )

(DEFINE-INSTANCE C24
  (:print-name "C24"
   :doc-string ""
   :is 3-PORT-CONNECTOR)
  (SOURCE-LIST (C24 Y0))
  (VAL 0)
  (CONN-1 Y0)
  (CONN-2 V9)
  (CONN-3 N9)
  )

(DEFINE-INSTANCE C25
  (:print-name "C25"
   :doc-string ""
   :is 3-PORT-CONNECTOR)
  (SOURCE-LIST
    (C25 A6 C20 A4 C17 N6 C15 V5 C36 N8 C22 A3 C12 A2
     C10 N3 C8 V4 C16 N5 C11 A1 C4 N1 C2 V1 C9 N4 W3 X3 C5
     N2 C1 Y4 C3 V2 W4 X4 C7 V3 C6 Y3 C13 Y2 C14 V6 W0 C19
     V7 C18 Y1 W1 X1))
  (VAL 1)
  (CONN-1 A6)
  (CONN-2 N9)
  (CONN-3 A7)
  )

(DEFINE-INSTANCE C26
  (:print-name "C26"
   :doc-string ""
   :is 3-PORT-CONNECTOR)
  (SOURCE-LIST
    (C26 A5 C22 A3 C16 N5 C14 V6 C23 N7 C18 Y1 C20 A4
     C17 N6 C12 A2 C10 N3 C8 V4 C15 V5 C13 Y2 W2 X2 C11 A1
     C4 N1 C2 V1 C9 N4 W3 X3 C5 N2 C1 Y4 C3 V2 W4 X4 C7 V3
     C6 Y3 C21 V8 W1 X1))
  (VAL 1)
  (CONN-1 A5)
  )

```

```

(CONN-2 A8)
(CONN-3 N10)
)

(DEFINE-INSTANCE C33
  (:print-name "C33"
   :doc-string ""
   :is 3-PORT-CONNECTOR)
  (SOURCE-LIST
    (C33 V11 C31 A7 C25 A6 C36 N8 C19 V7 C30 N10 W0 X0
     C26 A5 C22 A3 C16 N5 C14 V6 C23 N7 C18 Y1 C20 A4 C17 N6
     C12 A2 C10 N3 C8 V4 C15 V5 C13 Y2 W2 X2 C11 A1 C4 N1 C2
     V1 C9 N4 W3 X3 C5 N2 C1 Y4 C3 V2 W4 X4 C7 V3 C6 Y3 C21
     V8 W1 X1 C27 V9 C24 Y0))
  (VAL 0)
  (CONN-1 V11)
  (CONN-2 L2)
  (CONN-3 R1)
  )

(DEFINE-INSTANCE C34
  (:print-name "C34"
   :doc-string ""
   :is 3-PORT-CONNECTOR)
  (SOURCE-LIST
    (C34 V12 C32 A8 C26 A5 C23 N7 C21 V8 C29 N9 C25 A6
     C20 A4 C17 N6 C15 V5 C36 N8 C22 A3 C12 A2 C10 N3 C8 V4
     C16 N5 C11 A1 C4 N1 C2 V1 C9 N4 W3 X3 C5 N2 C1 Y4 C3 V2
     W4 X4 C7 V3 C6 Y3 C13 Y2 C14 V6 W2 X2 C19 V7 C18 Y1 W1
     X1 C28 V10 W0 X0 C24 Y0))
  (VAL 0)
  (CONN-1 V12)
  (CONN-2 L4)
  (CONN-3 R1)
  )

(DEFINE-INSTANCE W0
  (:print-name "W0"
   :doc-string ""
   :is 3-PORT-CONNECTOR)
  (SOURCE-LIST (W0 X0))
  (VAL 0)
  (CONN-1 X0)
  (CONN-2 N10)
  (CONN-3 V10)
  )

(DEFINE-INSTANCE W1
  (:print-name "W1"
   :doc-string ""
   :is 3-PORT-CONNECTOR)
  (SOURCE-LIST (W1 X1))

```

```

(VAL 0)
(CONN-1 X1)
(CONN-2 N8)
(CONN-3 V8)
)

(DEFINE-INSTANCE W2
  (:print-name "W2"
   :doc-string ""
   :is 3-PORT-CONNECTOR)
  (SOURCE-LIST (W2 X2))
  (VAL 1)
  (CONN-1 X2)
  (CONN-2 N6)
  (CONN-3 V6)
)

(DEFINE-INSTANCE W3
  (:print-name "W3"
   :doc-string ""
   :is 3-PORT-CONNECTOR)
  (SOURCE-LIST (W3 X3))
  (VAL 0)
  (CONN-1 X3)
  (CONN-2 V4)
  (CONN-3 N4)
)

(DEFINE-INSTANCE W4
  (:print-name "W4"
   :doc-string ""
   :is 3-PORT-CONNECTOR)
  (SOURCE-LIST (W4 X4))
  (VAL 0)
  (CONN-1 X4)
  (CONN-2 N1)
  (CONN-3 V2)
)

(DEFINE-INSTANCE A1
  (:print-name "A1"
   :doc-string ""
   :is ANDGATE)
  (SOURCE-LIST
    (A1 C4 N1 C2 V1 C9 N4 W3 X3 C5 N2 C1 Y4 C3 V2 W4 X4
     C7 V3 C6 Y3))
  (O-VAL 1)
  (I1-CONN C9)
  (I2-CONN C4)
  (O-CONN C11)
  (I2-VAL 1)
  (I1-VAL 1)
)

```

```

)

(DEFINE-INSTANCE A2
  (:print-name "A2"
   :doc-string ""
   :is ANDGATE)
  (SOURCE-LIST
    (A2 C10 N3 C8 V4 W3 X3 C6 Y3 C4 N1 C2 V1 C5 N2 C1 Y4
      C3 V2 W4 X4))
  (O-VAL 1)
  (I1-CONN C5)
  (I2-CONN C10)
  (O-CONN C12)
  (I2-VAL 1)
  (I1-VAL 1)
)

(DEFINE-INSTANCE A3
  (:print-name "A3"
   :doc-string ""
   :is ANDGATE)
  (SOURCE-LIST
    (A3 C12 A2 C10 N3 C8 V4 W3 X3 C6 Y3 C4 N1 C2 V1 C5
      N2 C1 Y4 C3 V2 W4 X4 C16 N5 C11 A1 C4 N1 C2 V1 C9 N4 W3
      X3 C5 N2 C1 Y4 C3 V2 W4 X4 C7 V3 C6 Y3 C13 Y2 C14 V6 W2
      X2))
  (O-VAL 1)
  (I1-CONN C12)
  (I2-CONN C16)
  (O-CONN C22)
  (I2-VAL 1)
  (I1-VAL 1)
)

(DEFINE-INSTANCE A4
  (:print-name "A4"
   :doc-string ""
   :is ANDGATE)
  (SOURCE-LIST
    (A4 C17 N6 C12 A2 C10 N3 C8 V4 C15 V5 C13 Y2 W2 X2
      C11 A1 C4 N1 C2 V1 C9 N4 W3 X3 C5 N2 C1 Y4 C3 V2 W4 X4
      C7 V3 C6 Y3))
  (O-VAL 1)
  (I1-CONN C11)
  (I2-CONN C17)
  (O-CONN C20)
  (I2-VAL 1)
  (I1-VAL 1)
)

(DEFINE-INSTANCE A5
  (:print-name "A5"

```

```

:doc-string ""
:is ANDGATE)
(SOURCE-LIST
  (A5 C22 A3 C12 A2 C10 N3 C8 V4 C16 N5 C11 A1 C4 N1
   C2 V1 C9 N4 W3 X3 C5 N2 C1 Y4 C3 V2 W4 X4 C7 V3 C6 Y3
   C13 Y2 C14 V6 W2 X2 C23 N7 C18 Y1 C20 A4 C17 N6 C12 A2
   C10 N3 C8 V4 C15 V5 C13 Y2 W2 X2 C11 A1 C4 N1 C2 V1 C9
   W3 X3 C5 N2 C1 Y4 C3 V2 W4 X4 C7 V3 C6 Y3 C21 V8 W1
   X1))
(O-VAL 1)
(I1-CONN C22)
(I2-CONN C23)
(O-CONN C26)
(I2-VAL 1)
(I1-VAL 1)
)

(DEFINE-INSTANCE A6
  (:print-name "A6"
   :doc-string ""
   :is ANDGATE)
  (SOURCE-LIST
    (A6 C20 A4 C17 N6 C12 A2 C10 N3 C8 V4 C15 V5 C13 Y2
     W2 X2 C11 A1 C4 N1 C2 V1 C9 N4 W3 X3 C5 N2 C1 Y4 C3 V2
     W4 X4 C7 V3 C6 Y3 C36 N8 C22 A3 C12 A2 C10 N3 C8 V4 C16
     N5 C11 A1 C4 N1 C2 V1 C9 N4 W3 X3 C5 N2 C1 Y4 C3 V2 W4
     X4 C7 V3 C6 Y3 C13 Y2 C14 V6 W2 X2 C19 V7 C18 Y1 W1
     X1))
    (O-VAL 1)
    (I1-CONN C20)
    (I2-CONN C36)
    (O-CONN C25)
    (I2-VAL 1)
    (I1-VAL 1)
  )

(DEFINE-INSTANCE A7
  (:print-name "A7"
   :doc-string ""
   :is ANDGATE)
  (SOURCE-LIST
    (A7 C25 A6 C20 A4 C17 N6 C15 V5 C36 N8 C22 A3 C12 A2
     C10 N3 C8 V4 C16 N5 C11 A1 C4 N1 C2 V1 C9 N4 W3 X3 C5
     N2 C1 Y4 C3 V2 W4 X4 C7 V3 C6 Y3 C13 Y2 C14 V6 W2 X2
     C19 V7 C18 Y1 W1 X1 C30 N10 W0 X0 C26 A5 C22 A3 C16 N5
     C14 V6 C23 N7 C18 Y1 C20 A4 C17 N6 C12 A2 C10 N3 C8 V4
     C15 V5 C13 Y2 W2 X2 C11 A1 C4 N1 C2 V1 C9 N4 W3 X3 C5
     N2 C1 Y4 C3 V2 W4 X4 C7 V3 C6 Y3 C21 V8 W1 X1 C27 V9
     C24 Y0))
    (O-VAL 1)
    (I1-CONN C25)
    (I2-CONN C30)
  )

```



```

(O-CONN C31)
(I2-VAL 1)
(I1-VAL 1)
)

(DEFINE-INSTANCE A8
  (:print-name "A8"
   :doc-string ""
   :is ANDGATE)
  (SOURCE-LIST
    (A8 C26 A5 C22 A3 C16 N5 C14 V6 C23 N7 C18 Y1 C20 A4
      C17 N6 C12 A2 C10 N3 C8 V4 C15 V5 C13 Y2 W2 X2 C11 A1
      C4 N1 C2 V1 C9 N4 W3 X3 C5 N2 C1 Y4 C3 V2 W4 X4 C7 V3
      C6 Y3 C21 V8 W1 X1 C29 N9 C25 A6 C20 A4 C17 N6 C15 V5
      C36 N8 C22 A3 C12 A2 C10 N3 C8 V4 C16 N5 C11 A1 C4 N1
      C2 V1 C9 N4 W3 X3 C5 N2 C1 Y4 C3 V2 W4 X4 C7 V3 C6 Y3
      C13 Y2 C14 V6 W2 X2 C19 V7 C18 Y1 W1 X1 C28 V10 W0 X0
      C24 Y0))
  (O-VAL 1)
  (I1-CONN C26)
  (I2-CONN C29)
  (O-CONN C32)
  (I2-VAL 1)
  (I1-VAL 1)
  )

(DEFINE-INSTANCE N1
  (:print-name "N1"
   :doc-string ""
   :is NANDGATE)
  (SOURCE-LIST (N1 W4 X4 C2 V1 C1 Y4))
  (O-VAL 1)
  (I1-CONN C2)
  (I2-CONN W4)
  (O-CONN C4)
  (I2-VAL 0)
  (I1-VAL 1)
  )

(DEFINE-INSTANCE N2
  (:print-name "N2"
   :doc-string ""
   :is NANDGATE)
  (SOURCE-LIST (N2 C1 Y4 C3 V2 W4 X4))
  (O-VAL 1)
  (I1-CONN C1)
  (I2-CONN C3)
  (O-CONN C5)
  (I2-VAL 1)
  (I1-VAL 0)
  )

```

```

(DEFINE-INSTANCE N3
  (:print-name "N3"
   :doc-string ""
   :is NANDGATE-3I)
  (SOURCE-LIST
    (N3 C8 V4 W3 X3 C6 Y3 C4 N1 W4 X4 C2 V1 C1 Y4))
  (I1-CONN C4)
  (I2-CONN C6)
  (I3-CONN C8)
  (I1-VAL 1)
  (I2-VAL 0)
  (I3-VAL 1)
  (O-CONN C10)
  (O-VAL 1)
  )

(DEFINE-INSTANCE N4
  (:print-name "N4"
   :doc-string ""
   :is NANDGATE-3I)
  (SOURCE-LIST
    (N4 W3 X3 C5 N2 C1 Y4 C3 V2 W4 X4 C7 V3 C6 Y3))
  (I1-CONN W3)
  (I2-CONN C5)
  (I3-CONN C7)
  (I1-VAL 0)
  (I2-VAL 1)
  (I3-VAL 1)
  (O-CONN C9)
  (O-VAL 1)
  )

(DEFINE-INSTANCE N5
  (:print-name "N5"
   :doc-string ""
   :is NANDGATE-3I)
  (SOURCE-LIST
    (N5 C11 A1 C4 N1 C2 V1 C9 N4 W3 X3 C5 N2 C1 Y4 C3 V2
      W4 X4 C7 V3 C6 Y3 C13 Y2 C14 V6 W2 X2))
  (I1-CONN C11)
  (I2-CONN C13)
  (I3-CONN C14)
  (I1-VAL 1)
  (I2-VAL 1)
  (I3-VAL 0)
  (O-CONN C16)
  (O-VAL 1)
  )

(DEFINE-INSTANCE N6
  (:print-name "N6"
   :doc-string ""

```

```

      :is NANDGATE-3I)
(SOURCE-LIST
  (N6 C12 A2 C10 N3 C8 V4 W3 X3 C6 Y3 C4 N1 C2 V1 C5
    N2 C1 Y4 C3 V2 W4 X4 C15 V5 C13 Y2 W2 X2))
(I1-CONN W2)
(I2-CONN C15)
(I3-CONN C12)
(I1-VAL 1)
(I2-VAL 0)
(I3-VAL 1)
(O-CONN C17)
(O-VAL 1)
)

(DEFINE-INSTANCE N7
  (:print-name "N7"
   :doc-string ""
   :is NANDGATE-3I)
  (SOURCE-LIST
    (N7 C18 Y1 C20 A4 C17 N6 C12 A2 C10 N3 C8 V4 C15 V5
      C13 Y2 W2 X2 C11 A1 C4 N1 C2 V1 C9 N4 W3 X3 C5 N2 C1 Y4
      C3 V2 W4 X4 C7 V3 C6 Y3 C21 V8 W1 X1))
    (I1-CONN C18)
    (I2-CONN C20)
    (I3-CONN C21)
    (I1-VAL 0)
    (I2-VAL 1)
    (I3-VAL 1)
    (O-CONN C23)
    (O-VAL 1)
  )

(DEFINE-INSTANCE N8
  (:print-name "N8"
   :doc-string ""
   :is NANDGATE-3I)
  (SOURCE-LIST
    (N8 C22 A3 C12 A2 C10 N3 C8 V4 C16 N5 C11 A1 C4 N1
      C2 V1 C9 N4 W3 X3 C5 N2 C1 Y4 C3 V2 W4 X4 C7 V3 C6 Y3
      C13 Y2 C14 V6 W2 X2 C19 V7 C18 Y1 W1 X1))
    (I1-CONN W1)
    (I2-CONN C19)
    (I3-CONN C22)
    (I1-VAL 0)
    (I2-VAL 1)
    (I3-VAL 1)
    (O-CONN C36)
    (O-VAL 1)
  )

(DEFINE-INSTANCE N9
  (:print-name "N9"

```

```

:doc-string ""
:is NANDGATE-3I)
(SOURCE-LIST
  (N9 C25 A6 C20 A4 C17 N6 C15 V5 C36 N8 C22 A3 C12 A2
   C10 N3 C8 V4 C16 N5 C11 A1 C4 N1 C2 V1 C9 N4 W3 X3 C5
   N2 C1 Y4 C3 V2 W4 X4 C7 V3 C6 Y3 C13 Y2 C14 V6 W2 X2
   C19 V7 C18 Y1 W1 X1 C28 V10 W0 X0 C24 Y0))
(I1-CONN C24)
(I2-CONN C25)
(I3-CONN C28)
(I1-VAL 0)
(I2-VAL 1)
(I3-VAL 1)
(O-CONN C29)
(O-VAL 1)
)

(DEFINE-INSTANCE N10
  (:print-name "N10"
   :doc-string ""
   :is NANDGATE-3I)
  (SOURCE-LIST
    (N10 W0 X0 C26 A5 C22 A3 C16 N5 C14 V6 C23 N7 C18 Y1
     C20 A4 C17 N6 C12 A2 C10 N3 C8 V4 C15 V5 C13 Y2 W2 X2
     C11 A1 C4 N1 C2 V1 C9 N4 W3 X3 C5 N2 C1 Y4 C3 V2 W4 X4
     C7 V3 C6 Y3 C21 V8 W1 X1 C27 V9 C24 Y0))
    (I1-CONN W0)
    (I2-CONN C27)
    (I3-CONN C26)
    (I1-VAL 0)
    (I2-VAL 1)
    (I3-VAL 1)
    (O-CONN C30)
    (O-VAL 1)
  )

(DEFINE-INSTANCE R1
  (:print-name "R1"
   :doc-string ""
   :is NORGATE)
  (SOURCE-LIST
    (R1 C33 V11 C31 A7 C25 A6 C36 N8 C19 V7 C30 N10 W0
     X0 C26 A5 C22 A3 C16 N5 C14 V6 C23 N7 C18 Y1 C20 A4 C17
     N6 C12 A2 C10 N3 C8 V4 C15 V5 C13 Y2 W2 X2 C11 A1 C4 N1
     C2 V1 C9 N4 W3 X3 C5 N2 C1 Y4 C3 V2 W4 X4 C7 V3 C6 Y3
     C21 V8 W1 X1 C27 V9 C24 Y0 C34 V12 C32 A8 C26 A5 C23 N7
     C21 V8 C29 N9 C25 A6 C20 A4 C17 N6 C15 V5 C36 N8 C22 A3
     C12 A2 C10 N3 C8 V4 C16 N5 C11 A1 C4 N1 C2 V1 C9 N4 W3
     X3 C5 N2 C1 Y4 C3 V2 W4 X4 C7 V3 C6 Y3 C13 Y2 C14 V6 W2
     X2 C19 V7 C18 Y1 W1 X1 C28 V10 W0 X0 C24 Y0))
    (O-VAL 1)
    (I1-CONN C33)
  )

```

```

(I2-CONN C34)
(O-CONN C35)
(I2-VAL 0)
(I1-VAL 0)
)

(Make-sponsor 'VALUE-PROPAGATION-RULES
:doc-string ""
:super-sponsor 'TOP-SPONSOR
:state :ENABLED)

(Make-sponsor 'PORT-RULES
:doc-string ""
:super-sponsor 'VALUE-PROPAGATION-RULES
:state :ENABLED)

(Make-sponsor 'BOARD-PORT-RULES
:doc-string ""
:super-sponsor 'VALUE-PROPAGATION-RULES
:state :ENABLED)

(Make-sponsor 'FDS-RULES
:doc-string ""
:super-sponsor 'TOP-SPONSOR
:state :ENABLED)

(DEFINE-RULE DIAGNOSE-NEXT-RULE
(:print-name "DIAGNOSE-NEXT-RULE"
:doc-string ""
:dependency NIL
:direction :FORWARD
:certainty 1.0
:explanation-string ""
:priority 0
:sponsor TOP-SPONSOR)
(INSTANCE DIAGNOSIS-SESSION IS USER-INTERFACE
WITH STATUS :DIAGNOSE-NEXT)
THEN
(INSTANCE DIAGNOSIS-SESSION IS USER-INTERFACE
WITH STATUS :NO)
(EVALUATE (DIAGNOSE-NEXT)))

(DEFINE-RULE 1-1JUNCTION-IN-RULE
(:print-name "1-1JUNCTION-IN-RULE"
:doc-string ""
:dependency NIL
:direction :FORWARD
:certainty 1.0
:explanation-string ""
:priority 500
:sponsor PORT-RULES)
(INSTANCE ?X IS 1-1JUNCTION

```

```

        WITH I-CONN ?Y)
      (INSTANCE ?Y IS CONNECTOR
        WITH VAL ?V)
    THEN
      (INSTANCE ?X IS 1-1JUNCTION
        WITH I-VAL ?V))

(DEFINE-RULE 3-1JUNCTION-OUT-RULE
  (:print-name "3-1JUNCTION-OUT-RULE"
    :doc-string ""
    :dependency NIL
    :direction :FORWARD
    :certainty 1.0
    :explanation-string ""
    :priority 500
    :sponsor PORT-RULES)
  (INSTANCE ?X IS 3-1JUNCTION
    WITH O-CONN ?Y
    WITH O-VAL ?V)
  THEN
    (INSTANCE ?Y IS CONNECTOR
      WITH VAL ?V))

(DEFINE-RULE 1-1JUNCTION-OUT-RULE
  (:print-name "1-1JUNCTION-OUT-RULE"
    :doc-string ""
    :dependency NIL
    :direction :FORWARD
    :certainty 1.0
    :explanation-string ""
    :priority 500
    :sponsor PORT-RULES)
  (INSTANCE ?X IS 1-1JUNCTION
    WITH O-CONN ?Y
    WITH O-VAL ?V)
  THEN
    (INSTANCE ?Y IS CONNECTOR
      WITH VAL ?V))

(DEFINE-RULE 2-1JUNCTION-OUT-RULE
  (:print-name "2-1JUNCTION-OUT-RULE"
    :doc-string ""
    :dependency NIL
    :direction :FORWARD
    :certainty 1.0
    :explanation-string ""
    :priority 500
    :sponsor PORT-RULES)
  (INSTANCE ?X IS 2-1JUNCTION
    WITH O-CONN ?Y
    WITH O-VAL ?V)
  THEN

```

```

(INSTANCE ?Y IS CONNECTOR
  WITH VAL ?V))

(DEFINE-RULE 3-1JUNCTION-IN-RULE3
  (:print-name "3-1JUNCTION-IN-RULE3"
   :doc-string ""
   :dependency NIL
   :direction :FORWARD
   :certainty 1.0
   :explanation-string ""
   :priority 500
   :sponsor PORT-RULES)
  (INSTANCE ?X IS 3-1JUNCTION
    WITH I3-CONN ?Y)
  (INSTANCE ?Y IS CONNECTOR
    WITH VAL ?V)
  THEN
    (INSTANCE ?X IS 3-1JUNCTION
      WITH I3-VAL ?V))

(DEFINE-RULE 3-1JUNCTION-IN-RULE2
  (:print-name "3-1JUNCTION-IN-RULE2"
   :doc-string ""
   :dependency NIL
   :direction :FORWARD
   :certainty 1.0
   :explanation-string ""
   :priority 500
   :sponsor PORT-RULES)
  (INSTANCE ?X IS 3-1JUNCTION
    WITH I2-CONN ?Y)
  (INSTANCE ?Y IS CONNECTOR
    WITH VAL ?V)
  THEN
    (INSTANCE ?X IS 3-1JUNCTION
      WITH I2-VAL ?V))

(DEFINE-RULE 3-1JUNCTION-IN-RULE1
  (:print-name "3-1JUNCTION-IN-RULE1"
   :doc-string ""
   :dependency NIL
   :direction :FORWARD
   :certainty 1.0
   :explanation-string ""
   :priority 500
   :sponsor PORT-RULES)
  (INSTANCE ?X IS 3-1JUNCTION
    WITH I1-CONN ?Y)
  (INSTANCE ?Y IS CONNECTOR
    WITH VAL ?V)
  THEN
    (INSTANCE ?X IS 3-1JUNCTION

```

```

        WITH I1-VAL ?V))

(DEFINE-RULE 2-1JUNCTION-IN-RULE2
  (:print-name "2-1JUNCTION-IN-RULE2"
   :doc-string ""
   :dependency NIL
   :direction :FORWARD
   :certainty 1.0
   :explanation-string ""
   :priority 500
   :sponsor PORT-RULES)
  (INSTANCE ?X IS 2-1JUNCTION
    WITH I2-CONN ?Y)
  (INSTANCE ?Y IS CONNECTOR
    WITH VAL ?V)
  THEN
    (INSTANCE ?X IS 2-1JUNCTION
      WITH I2-VAL ?V))

(DEFINE-RULE 2-1JUNCTION-IN-RULE1
  (:print-name "2-1JUNCTION-IN-RULE1"
   :doc-string ""
   :dependency NIL
   :direction :FORWARD
   :certainty 1.0
   :explanation-string ""
   :priority 500
   :sponsor PORT-RULES)
  (INSTANCE ?X IS 2-1JUNCTION
    WITH I1-CONN ?Y)
  (INSTANCE ?Y IS CONNECTOR
    WITH VAL ?V)
  THEN
    (INSTANCE ?X IS 2-1JUNCTION
      WITH I1-VAL ?V))

(DEFINE-RULE CREATE-INPUTSET-RULE
  (:print-name "CREATE-INPUTSET-RULE"
   :doc-string ""
   :dependency NIL
   :direction :FORWARD
   :certainty 1.0
   :explanation-string ""
   :priority 950
   :sponsor FDS-RULES)
  (INSTANCE DIAGNOSIS-SESSION IS USER-INTERFACE
    WITH STATUS :CREATE-INPUTS)
  THEN
    (INSTANCE DIAGNOSIS-SESSION IS USER-INTERFACE
      WITH STATUS :NO)
    (EVALUATE (CREATE-INPUTSET)))

```



```

(DEFINE-RULE BOARD-IMPORT-RULE
  (:print-name "BOARD-IMPORT-RULE"
   :doc-string ""
   :dependency NIL
   :direction :FORWARD
   :certainty 1.0
   :explanation-string ""
   :priority 500
   :sponsor BOARD-PORT-RULES)
  (INSTANCE ?X IS BOARD-INPUT-PORT
    WITH CONN ?Y
    WITH VAL ?V)
  THEN
    (INSTANCE ?Y IS CONNECTOR
      WITH VAL ?V))

(DEFINE-RULE START-WITH-WELCOME-SCREEN
  (:print-name "START-WITH-WELCOME-SCREEN"
   :doc-string "This causes the welcome-screen to be displayed when the FAULT-DIAGNOSIS-SYSTEM is :started"
   :dependency NIL
   :direction :FORWARD
   :certainty 1.0
   :explanation-string ""
   :priority 950
   :sponsor FDS-RULES)
  (INSTANCE FAULT-DIAGNOSIS-SYSTEM IS SCREEN-CONTROL
    WITH STATUS :STARTED)
  THEN
    (INSTANCE DIAGNOSIS-SESSION IS USER-INTERFACE
      WITH STATUS :NO)
    (INSTANCE FAULT-DIAGNOSIS-SYSTEM IS SCREEN-CONTROL
      WITH STATUS :RUNNING
      WITH NEW-SCREEN WELCOME-SCREEN))

(DEFINE-RULE CREATE-MODULE-RULE
  (:print-name "CREATE-MODULE-RULE"
   :doc-string ""
   :dependency NIL
   :direction :FORWARD
   :certainty 1.0
   :explanation-string ""
   :priority 900
   :sponsor FDS-RULES)
  (INSTANCE ACQUISITION-MENU IS POPUP-CHOOSE
    WITH ANSWER ?MOD-CLASS)
  THEN
    (EVALUATE (CREATE-MODULE-INSTANCE (CADR ?MOD-CLASS))))

(DEFINE-RULE CLEAR-INSTANCE-RULE
  (:print-name "CLEAR-INSTANCE-RULE"
   :doc-string ""

```

```

:dependency NIL
:direction :FORWARD
:certainty 1.0
:explanation-string ""
:priority 900
:sponsor FDS-RULES)
(INSTANCE DIAGNOSIS-SESSION IS USER-INTERFACE
  WITH STATUS :CLEAR)
THEN
  (INSTANCE DIAGNOSIS-SESSION IS USER-INTERFACE
    WITH STATUS :NO)
  (EVALUATE (CLEAR-CIRCUIT-INSTANCES)))

(DEFINE-RULE CLEAR-SINGLE-RULE
  (:print-name "CLEAR-SINGLE-RULE"
    :doc-string ""
    :dependency NIL
    :direction :FORWARD
    :certainty 1.0
    :explanation-string ""
    :priority 950
    :sponsor FDS-RULES)
  (INSTANCE DIAGNOSIS-SESSION IS USER-INTERFACE
    WITH STATUS :CLEAR-SINGLE)
  THEN
    (EVALUATE (CLEAR-SINGLE-MODULE)))

(DEFINE-RULE LIST-MODULES-RULE
  (:print-name "LIST-MODULES-RULE"
    :doc-string ""
    :dependency NIL
    :direction :FORWARD
    :certainty 1.0
    :explanation-string ""
    :priority 950
    :sponsor FDS-RULES)
  (INSTANCE DIAGNOSIS-SESSION IS USER-INTERFACE
    WITH STATUS :VIEW)
  THEN
    (INSTANCE DIAGNOSIS-SESSION IS USER-INTERFACE
      WITH STATUS :NO)
    (EVALUATE (LIST-VIEW-MODULES)))

(DEFINE-RULE DISPLAY-MODULE-RULE
  (:print-name "DISPLAY-MODULE-RULE"
    :doc-string ""
    :dependency NIL
    :direction :FORWARD
    :certainty 1.0
    :explanation-string ""
    :priority 900
    :sponsor FDS-RULES)

```

```

      (INSTANCE DIAGNOSIS-SESSION IS USER-INTERFACE
        WITH DISPLAY-MODULE ?MODNAME)
    THEN
      (INSTANCE DIAGNOSIS-SESSION IS USER-INTERFACE
        WITH INSTRUCTIONS ("Circuit Module: " ?MODNAME)
        WITH STATUS :NO)
      (EVALUATE (DISPLAY-MODULE-METHOD ?MODNAME)))

(DEFINE-RULE APPLY-INPUTSET-RULE
  (:print-name "APPLY-INPUTSET-RULE"
   :doc-string ""
   :dependency NIL
   :direction :FORWARD
   :certainty 1.0
   :explanation-string ""
   :priority 950
   :sponsor FDS-RULES)
  (INSTANCE DIAGNOSIS-SESSION IS USER-INTERFACE
    WITH STATUS :APPLY-INPUTS)
  THEN
    (INSTANCE DIAGNOSIS-SESSION IS USER-INTERFACE
      WITH STATUS :NO)
    (EVALUATE (APPLY-INPUTSET)))

(DEFINE-RULE BEST-TEST-RULE
  (:print-name "BEST-TEST-RULE"
   :doc-string ""
   :dependency NIL
   :direction :FORWARD
   :certainty 1.0
   :explanation-string ""
   :priority 950
   :sponsor FDS-RULES)
  (INSTANCE DIAGNOSIS-SESSION IS USER-INTERFACE
    WITH STATUS :BEST-TEST)
  THEN
    (INSTANCE DIAGNOSIS-SESSION IS USER-INTERFACE
      WITH STATUS :NO)
    (EVALUATE (BEST-TEST)))

(DEFINE-RULE MAKE-TEST-RULE
  (:print-name "MAKE-TEST-RULE"
   :doc-string ""
   :dependency NIL
   :direction :FORWARD
   :certainty 1.0
   :explanation-string ""
   :priority 950
   :sponsor FDS-RULES)
  (INSTANCE DIAGNOSIS-SESSION IS USER-INTERFACE
    WITH STATUS :MAKE-TEST)
  THEN

```

```

(INSTANCE DIAGNOSIS-SESSION IS USER-INTERFACE
  WITH STATUS :NO)
(EVALUATE (MAKE-TEST)))

(DEFINE-RULE CHECK-OUTPUT-RULE
  (:print-name "CHECK-OUTPUT-RULE"
   :doc-string ""
   :dependency NIL
   :direction :FORWARD
   :certainty 1.0
   :explanation-string ""
   :priority 300
   :sponsor FDS-RULES)
  (INSTANCE DIAGNOSIS-SESSION IS USER-INTERFACE
    WITH STATUS :CHECK-OUTPUT)
  THEN
    (INSTANCE DIAGNOSIS-SESSION IS USER-INTERFACE
      WITH STATUS :NO)
    (EVALUATE (FIRE-ALL-RULES))
    (EVALUATE (INITIAL-INFERENCE-DATA)))

(DEFINE-RULE BOARD-OUTPORT-RULE
  (:print-name "BOARD-OUTPORT-RULE"
   :doc-string ""
   :dependency NIL
   :direction :FORWARD
   :certainty 1.0
   :explanation-string ""
   :priority 500
   :sponsor BOARD-PORT-RULES)
  (INSTANCE ?X IS BOARD-OUTPUT-PORT
    WITH CONN ?Y)
  (INSTANCE ?Y IS CONNECTOR
    WITH VAL ?V)
  THEN
    (INSTANCE ?X IS BOARD-OUTPUT-PORT
      WITH VAL ?V))

(DEFINE-RULE FORGET-ALL-RULE
  (:print-name "FORGET-ALL-RULE"
   :doc-string ""
   :dependency NIL
   :direction :FORWARD
   :certainty 1.0
   :priority 0
   :sponsor TOP-SPONSOR)
  (INSTANCE DIAGNOSIS-SESSION IS USER-INTERFACE
    WITH STATUS :FORGET-ALL)
  THEN
    (INSTANCE DIAGNOSIS-SESSION IS USER-INTERFACE
      WITH STATUS :NO) a
    (EVALUATE (FORGET-ALL)))

```